Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

DSSP21 Build 061102 User Guide

Mike Mackay

Defence R&D Canada – Atlantic

Technical Memorandum
DRDC Atlantic TM 2006-252
December 2006

Canada

This page intentionally left blank.

DSSP21 Build 061102 User Guide

M. Mackay

Defence R&D Canada – Atlantic

Technical Memorandum
DRDC Atlantic TM 2006–252
December 2006

Author

M. Mackay

Approved by

N. Pegg
Head, Warship Performance

Approved for release by

K. Foster
Chair, Document Review Committee

Abstract

DSSP21 is a semiempirically based computer code for evaluating the dynamics and manoeuverability of streamlined underwater vehicles including submarines. The purpose of this user guide is to facilitate preparation of the input files that are required for a definition of the vehicle and its systems, and for controlling the calculations and manoeuvering simulations to be done. It contains detailed descriptions of input syntax and options, together with brief discussions of the output produced and some background information on the code structure and algorithms.

Résumé

Le DSSP21 est un code informatique à base semi-empirique servant à évaluer la dynamique et la manœuvrabilité des vaisseaux subaquatiques effilés, dont les sous-marins. Ce guide d'utilisation vise à faciliter la préparation des fichiers qu'il faut entrer pour définir le vaisseau et ses systèmes, le contrôle des calculs et la simulation des manœuvres. Il contient une description détaillée de la syntaxe des données d'entrée et des options connexes, ainsi que quelques mots sur les données produites à la sortie et de l'information contextuelle sur la structure et les algorithmes du code.

This page intentionally left blank.

Executive Summary

Introduction

DRDC Submarine Simulation Program, version 2.1 (DSSP21), was developed to predict the manoeuverability of submarines and smaller underwater vehicles. It generalizes the capability of an earlier Oberon-specific code that was used to study the tactical and safety implications of modifications to the submarine or to operating procedures; these investigations included towed array operation, ballast loss, sonar dome replacement, and diving limit redefinition. DSSP21 can model a wide range of vehicles and estimate their manoeuvering characteristics for a broad spectrum of system studies and evaluation, even preliminary design. Its direct predecessor, DSSP20, was used for a systematic study of the effect of a hull plug (e.g., for Air Independent Power) on manoeuvering, and to support development of the CF Remote Minehunting System.

Significance

DSSP21 is still in development. However, the current edition of the code is being frozen and documented at this time since it is a useful tool which has already found application. The documentation will be updated as further progress with development is made.

Although input to the program initially appears complex because of a large number of options, defaults are provided that allow most vehicle types and manoeuvers to be defined with relatively few parameters. This user guide is primarily intended to facilitate preparation of the input, which comprises a definition of the vehicle and its systems, and definition of the hydrodynamic and manoeuvering simulations to be done. The latter include calculation of static and dynamic loads on a captive vehicle, hydrodynamic coefficient generation, and free vehicle manoeuvers. The guide is intended for both experienced and new users.

Future Development

Additional capabilities to be incorporated into future editions of DSSP21 include compensation and trim, and automatic generation of Manoeuvering Limitation Diagram curves. In the longer term, a hybrid calculation mode may be developed to take best advantage of available experimental data and CFD analysis.

M. Mackay, 2006, DSSP21 Build 061102 User Guide, DRDC Atlantic TM 2006–252. Defence R&D Canada – Atlantic.

Sommaire

Introduction

La version 2.1 du programme de simulation sous-marine de RDDC (DSSP21) a été développée afin de prévoir la manœuvrabilité des sous-marins et de vaisseaux subaquatiques plus petits. Ce programme étend les capacités de l'ancien code Oberon qui avait été utilisé pour étudier les répercussions sur la tactique et la sécurité qu'aurait toute modification apportée au sous-marin ou aux procédures d'opération. Ces études ont porté entre autres sur l'opération de réseaux remorqués, la perte de ballast, le remplacement du dôme sonar et la redéfinition des limites de plongée. Le DSSP21 peut émuler un grand éventail de vaisseaux et estimer leurs caractéristiques de manœuvrabilité pour un vaste corpus d'études de systèmes et d'évaluations et même pour l'étude préliminaire. La version antérieure, le DSSP20, a servi à une étude systématique des effets d'un bouchon de coque (p. ex. pour l'alimentation sans air) sur les manœuvres et au développement du Système de déminage télécommandé des FC.

Importance

Le développement du DSSP21 se poursuit. Cependant, la version actuelle du code a été figée pour qu'on puisse la documenter, car il s'agit d'un outil pratique à qui on a déjà trouvé des applications. La documentation sera mise à jour à mesure que le développement continuera de progresser.

Même si l'entrée de données dans le programme peut sembler complexe à prime abord à cause du grand nombre d'options, le système offre des valeurs par défaut qui permettent de définir la plupart des types de vaisseaux et de manœuvres avec relativement peu de paramètres. Ce guide d'utilisation vise surtout à faciliter la préparation des données d'entrée, qui comportent la définition du vaisseau et de ses systèmes et la définition des simulations hydrodynamiques et des simulations de manœuvres à faire. Ces dernières comprennent le calcul des charges statiques et dynamiques sur vaisseau captif, la génération des coefficients hydrodynamiques et les manœuvres de vaisseau libre. Le guide s'adresse à la fois aux utilisateurs expérimentés et aux nouveaux utilisateurs.

Projets de développement

Des fonctions additionnelles seront ajoutées aux versions à venir du DSSP21, notamment la compensation et l'assiette, et la génération automatique des courbes des schémas de limitation des manœuvres. À plus long terme, on songe à développer un mode de calcul hybride pour tirer le maximum des données expérimentales et de l'analyse de la dynamique des fluides numérique.

M. Mackay, 2006, DSSP21 Build 061102 User Guide, DRDC Atlantic TM 2006–252. R&D pour la défence Canada – Atlantique.

Table of Contents

List of Figures

1 Introduction

DRDC Submarine Simulation Program, version 2.1 (`DSSP21`), is a code for evaluating underwater vehicle hydrodynamics and manoeuverability. The key calculations of hydrodynamic loads on the vehicle are done with component based semiempirical methods that are sufficiently accurate for at least preliminary design and routine manoeuvering analysis. Investigations are continuing to further improve these methods and widen their scope of application.

Developmental status of the code is indicated by a build number consisting of the release date in *yymmdd* format. In output from the program, an additional character may be appended: `A`, `B`, or `X`, indicating that the build is considered to be a preliminary, limited, or experimental release respectively. After sufficient validation the qualifier will be dropped. This guide applies to build 061102; subsequent builds will be accompanied by updates and additions to the guide as required. Changes from previously benchmarked editions of the code are summarized in annex A.

`DSSP21` proceeds in two phases:

- Geometry and systems setup. The user must provide a file defining the vehicle geometry and systems. The data is extensively checked, and as much preprocessing of it done as possible. This includes, for example, estimation of hydrodynamic interference effects, inertial and added masses and moments, and propulsor characteristics.
- Hydrodynamic and manoeuvering simulations. The simulations to be run are defined in sequence on an optional input file. Those currently implemented include static and dynamic loads on a captive vehicle, hydrodynamic coefficient generation, and free vehicle manoeuvering. If the simulation input file is absent, there is no work done in the second phase.

The principal objective in developing a multifunctional code was to base all calculations and simulations on a common set of hydrodynamic modeling subroutines. The underlying hydrodynamic models are therefore consistent, and improvements and enhancements to the models will be applied universally. In the previous version of the code, `DSSP20` [1,2], the same objective was achieved by distributing the different calculations and simulations between a number of individual programs which shared the core set of hydrodynamic subroutines. It was subsequently concluded that certain overheads incurred by this scheme were not justified, and that future maintainability would be better served by a single multifunctional program.

`DSSP21` is programmed in Fortran. The source code was revised to compile under Fortran 95 syntax rules; only a handful of features will not compile under Fortran 77. The library routine required to extract the optional command line parameter is compiler-dependent. `DSSP21` was developed on a Microsoft Windows PC using the Salford FTN95 Win32 version 2.00 compiler (Salford Software Ltd., `www.salfordsoftware.co.uk`),

and checked with STI's Understand for Fortran, version 1.4 (Scientific Toolworks Inc., `www.scitools.com`). Although the code is virtually platform-independent, this guide assumes a Windows platform where relevant.

The main function of this user guide is to facilitate preparation of the input files. Minimum input requirements are quite simple, but optional parameters are available to accommodate a wide range of vehicle configurations and to fine-tune the simulations. The hydrodynamic models and other aspects of the program are mostly described elsewhere; they are discussed here in only sufficient detail as required to define the input, and will be more fully described in a companion document to be published later.

New users are advised to first read this guide completely, with particular attention to section 3, General Considerations, and to the examples throughout the text. More experienced users will likely only need to refer to the input record descriptions in sections 4 to 22.

2 The DSSP21 Component Based Model

The program uses a breakdown of the vehicle into a small number of components whose individual and interactive hydrodynamics can be estimated or measured. Since for the most part it has a physical basis, this may be characterized as a rational flow approach.

For two components, subscripted 1 and 2, the hydrodynamic load vector $\mathbf{F}$ as a function of the extended state vector $\mathbf{u} = (u, v, w, p, q, r, \phi, \theta, \psi, \delta, \ldots)^T$ — where u, v, w, p, q, r are translational and rotational velocities, ϕ, θ, ψ are the Euler angles, and $\delta, \ldots$ represents control surface deflections and other parameters — is expressed as:

$$\mathbf{F}(\mathbf{u}) = \mathbf{C}_1(\mathbf{u}) + \mathbf{C}_2(\mathbf{u}) + \mathbf{C}_{1,2}(\mathbf{u}) + \mathbf{C}_{2,1}(\mathbf{u}) \tag{1}$$

in which $\mathbf{C}_1(\mathbf{u})$ and $\mathbf{C}_2(\mathbf{u})$ are the individual loads on each component and $\mathbf{C}_{1,2}(\mathbf{u})$ and $\mathbf{C}_{2,1}(\mathbf{u})$ are the interactions of each with the other. It is implied that each of the terms in equation (1) is estimated or measured in a local axis system for local flow and then transformed into a common axis system for summation. In the present build, there are many more terms, but the scheme is still limited to two-component interactions:

$$\mathbf{F}(\mathbf{u}) = \sum_i \mathbf{C}_i(\mathbf{u}) + \sum_i \sum_{j, j \neq i} \mathbf{C}_{i,j}(\mathbf{u}) \tag{2}$$

The calculations that are implemented in `DSSP21` model streamlined hulls and low aspect ratio lifting surfaces. Parameters required to model hull-appendage interactions are included as part of the appendage data in the geometry input file (file nomenclature is described in section 3.1). This guide does not detail the load calculations; however, discussion of many of the algorithms can be found in references [3,4,5], and added mass calculations are discussed in reference [6].

To specify hydrodynamic and manoeuvering simulations, the user must provide a second input file. The present simulation options are:

- Static load calculation. Static loads, for example as measured in a captive model experiment, are calculated for defined values of the velocities, control deflections, and incidence angles.
- Coefficient generation. At present, the approach implemented in the `DERIVS` program [7] is employed. Stability derivatives and other hydrodynamic quantities are estimated from predefined static runs.
- Captive dynamic manoeuver simulation. Dynamic loads, for example as measured in a captive model experiment, are calculated from time histories of the velocities and control deflections.
- Free-running manoeuver simulation. The equations of motion and a number of auxiliary equations are integrated over time using an order (3,4) Runge-Kutta adaptive timestep routine with optional interpolation of the output to regular time increments [8,9].
- MLD curve generation. Not presently implemented; it will require a predefined series of free manoeuvers with pass/fail termination criteria.

3 General Considerations

3.1 File Nomenclature

The naming scheme for most `DSSP21` input and output files has the form *Root.Ext*. *Root* is a user-defined string up to eight characters long ('`dssp21`' is used by default) and *Ext* indicates the function of the file. In addition, each simulation may generate files `sim`*nnn*`.txt` and `aux`*nnn*`.txt`, where *nnn* is the number of the simulation in the input file sequence. An arbitrarily-named time history input file is required for a captive simulation.

Root may be input as a command line parameter so that the program can be run from another without any user intervention. Without a command line parameter, *Root* is requested from the user as described in section 23. In either case, *Root* is truncated to eight characters if it is longer.

File names are interpreted as follows:

File	Function
Root`.geo`	input file for geometry and systems setup
Root`.gpr`	geometry and systems printer output
Root`.gpl`	geometry plotting output

Root`.log`	run log output
Root`.sim`	simulation sequence input file
Root`.spr`	simulation sequence printer output
`sim`*nnn*`.txt`	tabulated output from simulation *nnn*
`aux`*nnn*`.txt`	tabulated auxiliary output from simulation *nnn*

The absence of *Root*`.geo` generates a fatal error. All input and output files consist of ASCII characters, and can be read and manipulated with a text editor. The tabular output files `sim`*nnn*`.txt` and `aux`*nnn*`.txt` can be opened directly by most spreadsheet and graphical analysis software.

3.2 User-Prepared Input

Input files *Root*`.geo` and *Root*`.sim` are prepared by the user with a text editor. The input character set for `DSSP21` comprises upper and lower case letters, the space, digits `0` to `9`, the 'printable' characters `*()-=+;:',<.>/?‘~!@#$%^&_\|[{]}"`, and the tab. Unrecognized characters generally result in a fatal error. Most input is not case-sensitive.

In order to allow these files to be more readable, a simple input parsing scheme is employed. Each line in the input file is treated as a separate record of up to 71 characters consisting of words separated by one or more delimiters and terminated by an optional end-of-record character. The word delimiters are the space, comma '`,`', semicolon '`;`', and tab. The end-of-record (EOR) characters are the double quote '`"`', exclamation point '`!`', and percentage sign '`%`'. A word is a string of any other characters. Input following an EOR character is ignored and may therefore constitute a comment. Any input that is encountered after the input requirements of a record are fulfilled is similarly ignored.

Input is controlled by keywords. A keyword consists of an alphabetic string of up to ten characters contained in one of the parser's dictionaries; listings are given in annex B. On input, keywords are not case-sensitive. Most input records consist of a keyword followed by one or more arguments. An argument may be numerical, some non-keyword text, or another keyword. Since some arguments are optional, it is prudent to always precede a following comment by an EOR character.

Unless noted, keywords discussed in this guide are contained in the main dictionary. These keywords may be abbreviated by truncation so long as the result is not ambiguous. For example, keywords `Left` and `Length` can be abbreviated by `Lef` and `Len` respectively, but `Le` is ambiguous. On the other hand, an unabbreviated keyword is distinguished from any abbreviations that may be identical: `Delta` and `DeltaLim` are therefore recognized as separate keywords. For this reason, keyword abbreviation should be used with care since the result may be unexpected. Finally, it should be noted that the same keyword may be used for different purposes in different parts of the input, its interpretation depending on the context.

With keyword controlled input, records can in general appear in any order in the input file. In DSSP21 this freedom is limited to within an input 'level': input is initially at Level 1 and is switched to Level 2 and higher by specific keywords. Moreover, input order is significant is within a data block, i.e., a sequence of physical records that is is treated as a single logical record. And some simulations, notably for static load estimation, execute records in the *Root*.sim file in input order, so that changing that order may affect the results.

Text records provide indentification and other information for printed output. They consist of the keyword Text, one or more delimiters, and a string of any characters in the DSSP21 character set. On output, case in a Text string is preserved.

A label string uniquely identifies a vehicle component. It consists of 1 to 24 alphabetic characters preceded by a label character, '#' or '?'. Labels are useful for clarifying the object of control commands, and may be defined by the user; they default to #HULL001, #LIFT001, #PROP001, etc. Labels are not case-sensitive.

Numerical values are also parsed. An integer input can have no more than 20 characters, and is read, right justifed, in Fortran I20 format. A floating-point input can have no more than 30 characters. Initially it is read, right justified, in I30 format and converted to a double precision variable; if that fails, it is re-read in F30.0 format into a double precision variable.

Blank lines in input are ignored, and purely comment lines should start with an EOR character otherwise the parser will try to interpret the first delimited string as a keyword.

The following are typical input records that illustrate a number of the concepts introduced above. They were picked arbitrarily and do not represent a continuous segment of data input:

```
Text Model UT at nominal full-scale
HELP
Reference           !(1) \
   -26.2  .0  .0    !(2)  | This is a data block
      .0  .0  .0    !(3) /
rho ; 1025
"This is a comment
Label #XRudderSevenThirty  % Labels are optional
PReverse  -49.8  .5e0  -.7d0
Interpl,,,5,,,TypeOne
WeightDCI 7  0 0 0
Loop Delta Port -15 15 1
WeightDCI All One
```

3.3 Error Messages

This guide is not, except in the broadest terms, concerned with the architecture of the code. However, some familiarity with it is useful because error messages cite the

program unit (i.e., subroutine, function, etc.) in which the error occured, thus providing a context for understanding the problem. A list of program units and their functions is given in annex C.

Error messages are issued to both the terminal and *Root*`.log`. They have four levels of severity: fatal, warning with pause, warning, and information. Fatal error messages are generated by a condition or problem that forces termination of program execution. A warning with pause indicates an actual or potential problem for which the user may wish to terminate program execution (the default response is to continue). Simple warnings are issued to indicate an actual or potential problem while allowing program execution to continue automatically. In a few instances, an information message is issued; this indicates that the program is flagging, at worst, a benign anomaly that the user should be aware of.

The general format for an error message is:

```
    !! Fatal Error in program_unit_name
or  ?? Warning from program_unit_name
or  -- Information from program_unit_name
        message line 1
        message line 2
```

If a warning includes a pause, it is followed by:

```
    ?? Continue execution ?  ...  [Y]/N
```

and the user's response is noted in the *Root*`.log` file as one of:

```
    ?! Execution terminated by user
or  ?? Execution continued by user
```

Message line 2 is to clarify the error report on line 1, and is only present if required. As the code was assembled, some error conditions have become trapped in higher-level routines, and corresponding lower level error checks have become redundant. Such errors are noted on line 2 as '`- impossible`'. Since a number of such errors may have been missed or misidentified, if this message is encountered please inform the author (at `mike.mackay@drdc-rddc.gc.ca`), providing input files and other supporting information. Other error messages may include the notation '`- system error`' on line 2, indicating that they originate from compiler- or platform-related errors. These are typically '`ERR=`' exits from a Fortran `READ`, `WRITE`, or `OPEN` command. The last, for example, may result from `DSSP21` attempting to open an output file that is already open in another program.

Syntax and other errors in user input should be straightforward to identify because the input is echoed record-by-record to the *Root*`.log` file, and in most cases the error message will be displayed just after the offending input record.

An example of each type of error message follows:

```
!! Fatal Error in GEO_L2_HUL
   Unsavable label
   - label string is duplicated or invalid

?? Warning from SIM_L2_CAP
   Default RHO =  1000.000 for the first sim
   - perhaps it should have been changed...
?? Continue execution ?  ...  [Y]/N

?? Warning from PRPCOM
   Invalid second argument
   - keyword not in dictionary

-- Information from GEO_L2_HUL
   HULFOR: number of stations not defined
   - default (21) assumed
```

A handful of fatal errors have a less obvious solution. We summarise some of these here, but the reader may need to refer forward in the guide to follow the explanations. Others are noted elsewhere in the text.

A truncated or very bluff hull afterbody may generate the following error:

```
!! Fatal Error in HULLH1_HUL4
   No minimum found in dA/dx (DT2) for separation
   - very bluff afterbody?  - force separation
```

indicating that the hull load algorithm has failed to find a minimum needed in the slope of the afterbody cross sectional area to locate flow separation. The user should check for irregularities in the afterbody offset data. If they appear satisfactory, the region of likely separation, if known, can be specified with a `HullSep` record (section 5.3).

The time integration in a free manoeuvering simulation may generate an error of the form:

```
!! Fatal Error in SPRINT
   Unexpected IND = N from DDNORK (interpolated/stepped output)
   - inconsistent timesteps and/or tolerance ??
```

where N is one of

- `-1`: an internal limit on the number of function evaluations has been exceeded; this condition should not be possible in `DSSP21`.
- `-2`: the current minimum timestep is greater than the maximum; this should not be possible in `DSSP21` unless these quantities have been input incorrectly with the `TimeStep` record, see section 22.3.
- `-3`: the integrator is unable to satisfy the accuracy required to continue. Possible causes of this are that integration tolerance is too small relative to the minimum timestep

or that the matrix of inertial plus added mass properties is ill-conditioned (sections 3.4 and 22.3). The latter can alternatively result in an unexpected stop with the message '`DYDT: sim stopped by zero/negative forward speed`'. The key here is that the stop is unexpected: the program stops with this message because u ceases to be positive (as would legitimately occur in simulating a crash-back, for example), but the prior time history is not consistent with this condition. If the problem is associated with a reference shift, it may be avoided by redefining the vehicle's target point instead of the reference (section 3.4).

Please report time integrator errors with $N =$ `-1` (and $N =$ `-2` if `TimeStep` appears correct) to the author, providing input files and other supporting information.

A captive model simulation may fail with the error '`TIMEHISTRY file P data is inconsistent`' (or '`...Q...`', or '`...R...`'). This generally indicates a fundamental error in the user-supplied state variable time histories, see section 21.3.

3.4 Axis Systems

The relationship between earth-fixed and body-fixed axis systems is discussed in standard textbooks, e.g., Etkin [10]. There are two principal body-fixed axis systems in `DSSP21`: vehicle axes, in which vehicle geometry and most other input is defined, and reference axes, in which output quantities such as hydrodynamic forces and moments are defined. Both systems are righthanded. Vehicle axes are determined indirectly by the (x, y, z) coordinates of vehicle geometry which are initially defined in file *Root*`.geo`. All components of the vehicle must use vehicle axes for their definition. Coordinates and lengths are in metres, and angles are in degrees.

Reference axes default to vehicle axes shifted to an origin at the hull CB of a single-hulled vehicle; for other configurations reference axes must be defined by the user. To be most meaningful, reference axes should coincide with conventional hydrodynamic axes, i.e., with x directed forward, y to starboard, and z downward. Likewise, but also for reasons of computational stability in a free manoeuver, the reference axis origin should be in the vicinity of the vehicle CB and CG.

The reference origin trajectory is always output in a free manouever, but the user may define an additional point, called the target, which is also tracked. This is typically a point of interest such as the attachment point for a towed array. Defining it as the target avoids numerical difficulties that could be encountered when shifting the reference origin to achieve the same result. (The problem is principally with the matrix comprising the inertial plus added mass properties of the submarine; it must be inverted to decouple the equations of motion for integration in time. Since internal `DSSP21` calculations are dimensioned, some elements of the matrix grow rapidly as the reference origin is moved away from the principal axes of the vehicle, making it ill-conditioned for inversion. Nondimensionalizing this part of the calculations may alleviate the problem, but that is outside the scope of current code development.)

In a free-running manoeuver there is a set of earth-fixed axes (x_0, y_0, z_0) that are parallel to the reference axes at the start of the simulation with their origin on the surface rather than at initial depth $z_{0\bullet}$. Consequently, $(x_0, y_0, z_0)_{\text{start}} = (0, 0, z_{0\bullet})$. where here, as elsewhere in this guide, subscript $\bullet$ indicates an initial condition. The trajectory of the reference origin in (x_0, y_0, z_0) coordinates is tabulated on file `sim`*nnn*`.txt`, which includes the state variables u, v, w, p, q, and r in the reference coordinate system..

Internally, the program uses a separate local axis system for each component. For a hull component the origin is at its longitudinal CB, local x is directed forward along the longitudinal axis, and y and z are determined from vehicle axes or are user-defined. For a lifting appendage the origin is at the root mid-chord, local y is outward along the mid-chord line, z is normal to the effective plane of the appendage, and x is obtained from the cross product of y and z. In a few instances, printed output is in local axes. The main reason for the user to be aware of them is that individual control deflections are defined in appendage local axes, positive for rotation of z into x.

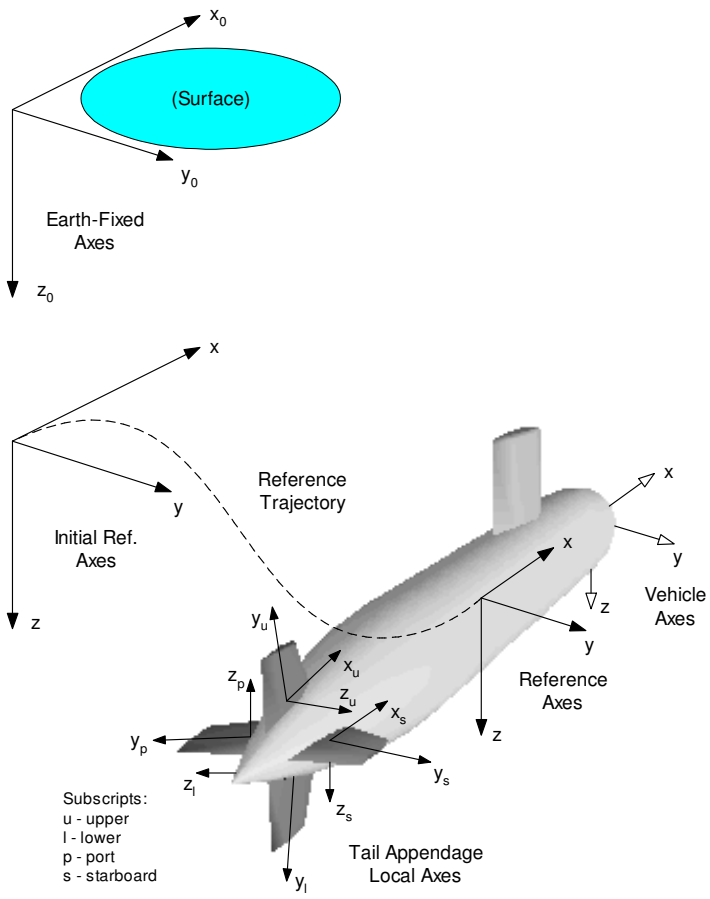


Figure 1. Principal earth-fixed and boat-fixed axis systems.

Examples of these axis systems are sketched in figure 1. In this case, the vehicle axis origin is at the bow of the submarine. For all-moving rudders on the same shaft, the definition of appendage local axes results in deflections of opposite sign on the upper and lower rudders; the same observation applies to other pairs of control planes.

With one exception, noted in section 4.2, all geometrical input must be given in the vehicle axis system. On the other hand, loads and other hydrodynamic quantities are output in reference axes.

There is another earth-fixed axis system to note in connection with free-running manoeuvers: chart axes (x_e, y_e, z_e). In this system x_e is directed North, y_e East, and z_e down. At present, they serve no purpose other than to locate the target point of the vehicle on a notional chart, but may be useful for future developments such as defining density as a spatial variable. In chart axes, the target initial position and heading are defined by `Origin` and `Heading` records (section 22.2) as illustrated in figure 2. Target coordinates (x_T, y_T, z_T) in chart axes are output on files *Root*`.spr` and `aux`*nnn*`.txt`; the latter also tabulates target translational velocities u_T, v_T, and w_T in reference axes.

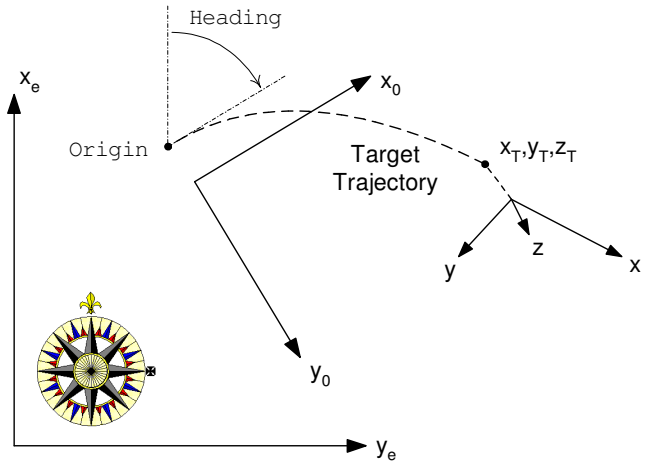


Figure 2. The target trajectory, (x_0, y_0, z_0) axes, and reference axes (x, y, z) relative to the chart axis system.

3.5 Applicability of the Program

It is implicit in many algorithms used in the current version of `DSSP21` that the vehicle is streamined and at moderate incidence, i.e., the vehicle components are reasonably slender and have their hydrodynamic axis — the longitudinal axis of a hull or the chordline of a lifting appendage — oriented more-or-less in the direction of motion (or against the direction of incident flow). The program cannot model bluff or irregular underwater vehicles such as ROVs, but can model submarines, many UUVs, and towfish.

Vehicle geometry is currently described by up to two hull components and up to twelve lifting appendages; there can be up to three propulsors.

4 Root.geo Level 1 Input

Root`.geo` input defines component geometry in simple terms, e.g., hull breadth and height. Since vehicle components are input independently, interference calculations need additional parameters such as the hull radius at the attachment point of an appendage. The locations and characteristics of propulsors and other systems are needed to complete the description. In this first phase of processing, hydrodynamic quantities are calculated for fresh water density, $\rho = 1000$ kg/m^3; later, they are scaled for a given density.

An example of *Root*`.geo` is given in annex D. Level 1 input is used to flag the vehicle components — descriptions of which are then input at Level 2 — and to provide overall program control. Return to Level 1 from Level 2 occurs automatically when an out-of-context keyword is encountered.

The following table summarizes the keyword records at level 1.

Keyword	Function
`AddedMass`	define method for added mass and moment calculations
`AutoDepth`	depth autopilot parameter input to follow
`AutoHead`	heading autopilot parameter input to follow
`AutoProp`	propulsion autopilot parameter input to follow
`AutoRoll`	roll autopilot parameter input to follow
`BG`	define **BG** (vector)
`CB`	define location of CB
`CG`	define location of CG
`Help`	send help output to *Root*`.log`
`HPAir`	HP air bottle group (BG) input to follow
`Hull`	hull component input to follow
`Inertial`	define method for calculation of inertial properties
`Lift`	lifting component input to follow
`Lump`	lump component input to follow (inactive)
`MBT`	main ballast tank (MBT) component input to follow
`OOPCalc`	out-of-plane load calculation input to follow
`Plot`	generate a geometry plot file
`Prop`	propulsor component input to follow
`Propulsor`	propulsor component input to follow
`Reference`	define reference origin and axes

`Text`	title or other information
`WagBInit`	initialize Wageningen B propellers
`WTC`	WTC component input to follow

`Reference` is required with no hull; otherwise these records are optional and may in general appear in any order on the file. One exception is that association between BGs and MBTs may be implied by the order of `HPAir` and `MBT` records (section 11).

In the record descriptions following, and elsewhere in this guide, optional arguments are enclosed in brackets, [ ].

4.1 AddedMass and Inertial Records

These records control the estimation, in reference coordinates, of added masses and added moments of inertia, and of inertial properties such as mass and moments of inertia. In each case, there is at present only one method of calculation available; these records provide placemarkers for possible future development.

`AddedMass` [*keyword*]

The argument may be one of `Esam` or `None`. `Esam` is the default. If the argument is `None`, the calculation is omitted; otherwise components of the `ESAM` code [6] are used, and portions of the internal `ESAM` dialogue are sent to *Root*`.log`.

`Inertial` [*keyword*]

The argument may be the one of `Form` or `None`. `Form` is the default. If the argument is `None`, the calculation is omitted; otherwise form displacement, assuming uniform density, is used as the basis for the estimates.

The resulting added mass and inertial matrices are output to *Root*`.gpr`.

4.2 BG, CB, and CG Records

The centers of buoyancy and gravity $\mathbf{CB} = (x_B, y_B, z_B)$ and $\mathbf{CG} = (x_G, y_G, z_G)$ are defined in vehicle axes, and the full vector form of the metacentric distance is expressed as $\mathbf{BG} = (\mathbf{CG} - \mathbf{CB})$. However, its normal component $z_{BG} = (z_G - z_B)$ is identified with the metacentric height $\overline{BG}$, and that, since it acts as a hydrodynamic quantity in various calculations, should be expressed in reference axes. Hence, **BG** is the only 'geometric' quantity to require input in the reference, rather than the vehicle, axis system.

The three records are:

`BG` $[x_{BG}\ ,\ y_{BG}]\ ,\ \overline{BG}$

Defines the components of **BG** in reference axes. A single argument is taken to be $\overline{BG}$ while the other two components default to zero. Otherwise, all three arguments must be present. $\overline{BG}$ must not be negative. If there is only one `Hull` component, $\overline{BG}$ defaults to $D_M/20$, where D_M is maximum diameter of the hull. See the discussion below on combining `BG`, `CB`, and `CG` records.

`CB` $x_B\ ,\ y_B\ ,\ z_B$

Defines the three components of **CB** in vehicle axes. Default values are derived from the form displacement.

`CG` $x_G\ ,\ y_G\ ,\ z_G$

Defines the three components of **CG** in vehicle axes. Default values are derived from the **BG** and **CB** defaults.

Locating CB and CG, and thereby determining **BG**, requires any two of the three records. Providing one or none causes the program to resort to the defaults, while providing all three will result in a fatal error if they are inconsistent.

4.3 Hull, Lift, and OOPCalc Records

Up to two `Hull` components and twelve `Lift` (i.e., appendage) components can be defined.

`Hull` [*keyword* , $[N_S]$]

Announces a new hull component. The first optional argument selects the method for calculating hull loads. It may be `HulFor`, indicating that the `HulFor` code [11] is to be used; or it may be `Default` or `HulFoM`, indicating that the empirical method of reference [12] is used to calculate translational loads and `HulFor` used to calculate rotational loads; anything else is ignored. The default is `HulFoM`. The second argument, N_S, which is only read if the first is present and recognized, is the number of hull stations to be used in the load calculation; $11 \leq N_S \leq 51$. By default $N_S = 21$, which is sufficient for typical submarine hulls.

The level 2 hull input records are described in section 5.

`Lift`

Announces a new lifting appendage component. The level 2 appendage input records are described in section 6.

`OOPCalc` [*keyword*]

If the optional keyword argument is `None`, this record suppresses all out-of-plane load calculations. Otherwise, it announces the specification for a new calculation according to the level 2 input records described in section 7. By

default, an out-of-plane load calculation is done for each lifting appendage with the type `Sail`.

4.4 Prop, Propulsor and WagBInit Records

Up to three propulsors can be defined. `Prop` is a synonym for `Propulsor`; making it a keyword allows the common abbreviation `Prop` to be used without ambiguity. Three different propulsor types, listed below, are currently implemented. The `WagBInit` record and associated level 2 input are required to override defaults used for an initial optimization of `WageningB`-type propulsors.

`Prop` [*keyword*]

Announces a new propulsor. The argument indicates propulsor type. If present, it must be one of: `DSSPtwo`, `STRprop`, or `WageningB`; the default is `WageningB`. Mixing propulsor types is not recommended; in any case, `WageningB` propulsors cannot be mixed with other types.

A type specifies the method of estimating propulsor thrust and torque coefficients K_T and K_Q. In `DSSP21`, these coefficients are derived as functions of the advance ratio based on ship speed, $J_S = u/nD_P$ (not incorporating the wake fraction). The three methods are:

`DSSPtwo`. This is a 'small main propulsor' propeller model approximating the model used in `DSSP02` [13] for the Oberon class; i.e., an older twin-screw submarine arrangement. The model is based on simple functions of the propulsion ratio n/n_0, where n_0 is self-propulsion RPM. It may not represent single screw propulsion particularly well since the propeller inflow conditions are different.

`STRprop`. This specifies a 'large main propulsor' propeller model based on the four-quadrant single screw behind-model wind tunnel data reported in reference [14]. The propulsive efficiency, $(K_T/K_Q) \cdot (J_S/2\pi)$, of this model is unrealistically high at large J_S.

`WageningB`. This specifies a 'large main propulsor' propeller model based on two-quadrant open water data for the Wageningen B4–70 series [15]. The method includes semiempirical calculations for wake fraction and thrust deduction [16], and approximates optimum pitch/diameter ratio.

Level 2 propulsor input records are described in section 8.

`Propulsor` [*keyword*]

As `Prop`, above.

`WagBInit`

Announces level 2 input for initializating `WageningB` propulsors; this input is described in section 9.

4.5 The Reference Record

This record is required to set (or override the default) reference axes and origin in *Root*`.geo` level 1. Furthermore, a new reference definition, which will supercede this one, can be supplied with simulation level 2 input in *Root*`.sim`. At any stage, it will avoid problems if the reference origin is in the vicinity of the vehicle CB and CG, and if the reference axes correspond to conventional hydrodynamic axes (section 3.4).

`Reference` [*keyword*]

If this record has the keyword `CB` or `CG` as an argument then there is no level 2 input and the reference origin is located at the CB or CG of the vehicle with the reference axes directed along the vehicle axes. Otherwise, the reference origin and axes (angles of rotation from vehicle axes, in order, in degrees) must follow as triplets of real numbers on a two-record data block:

x_o y_o z_o
ψ_o θ_o ϕ_o

4.6 Autopilot Records: AutoDepth, AutoHead, AutoProp, and AutoRoll

`AutoDepth` [*keyword*]
`AutoHead` [*keyword*]
`AutoProp` [*keyword*]
`AutoRoll` [*keyword*]

There are four autopilots available for manoeuvering simulations in `DSSP21`: for depth, heading, propulsion, and roll. They are provided primarily to simplify the task for the user; for example, to automatically maintain depth during a turn. The records listed above announce input of parameters for the appropriate autopilot. The optional keyword argument indicates a default autopilot type, see the discussion in section 10; it can be `PhaseLead` or `PID`. In the absence of this argument the depth, heading, and roll autopilots default to `PhaseLead`, and the propulsion autopilot to `PID`. Optional level 2 input records that follow are described in section 10. Use of the autopilots in a simulation is described in section 22.4.

4.7 HPAir, MBT, and WTC Records

Up to nine `HPAir` bottle group, eight `MBT`, and eight `WTC` components can be defined.

`HPAir`

Announces a new bottle group component. The level 2 `HPAir` input records are described in section 11.

`MBT`

Announces a new main ballast tank component. The level 2 `MBT` input records are described in section 11.

`WTC`

Announces a new watertight compartment component. The level 2 `WTC` input records are described in section 12.

4.8 Other Records: Help, Plot, and Text

`Help`

When a `Help` record is encountered, help text is sent to *Root*`.log`, and the program is stopped. The help text is intended to be a quick reference for the user if this guide is not readily available; its principal components are dictionary listings and a user input summary.

`Plot` $[r_{view}]$

Announces level 2 geometry plotting information as described in section 13. Without this record the plot file *Root*`.gpl` is not generated. The optional argument r_{view} determines the viewing distance for initially rendering VRML images in some viewing software; in the present build it is ignored for other plotting formats. By default r_{view} is the length of the hull, if there is only one; otherwise it is zero.

`Text` *text string*

This record comprises a string of up to 64 characters preceded by the keyword `Text` and one or more delimiters. The string is not parsed, and may therefore contain any character, including delimiters and EOR characters.

The function of a `Text` record is to provide title and other information for the run. Each one is output to *Root*`.gpr` as it is encountered, and the first ten `Text` records are saved as heading information for the simulation phase of program execution.

5 Root.geo Level 2 Hull Input

The nomenclature used to define basic hull component geometry is illustrated in figure 3. A hull is defined at a series of stations between the nose and tail. These data may be interpolated to a different set of stations. The minimum information required is diameter at each station for an axisymmetric hull, and a direct or indirect definition of the nose and tail points in vehicle axes. The tail section may be closed, with zero

diameter, or truncated, with a nonzero diameter. In the latter case, base drag can be modeled. Additional optional input for each station includes breadth and height (in the case of non-axisymmetry), cross section area, and camber (which is only used for plotting geometry in this version of the program).

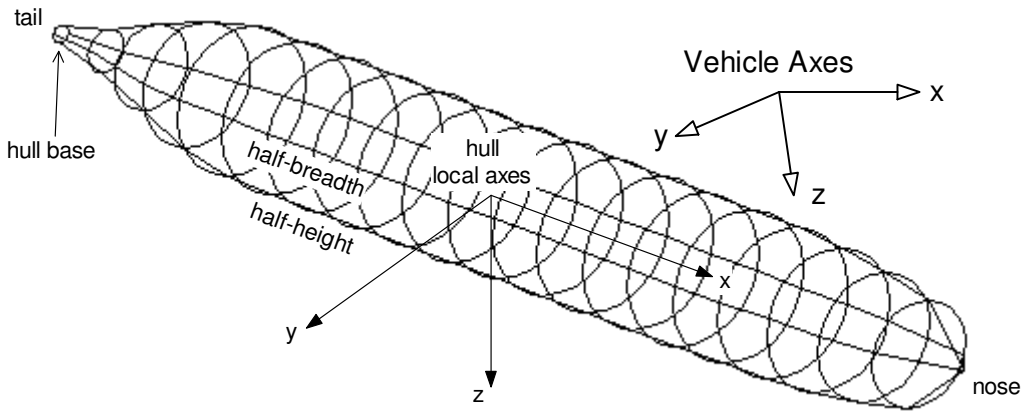


Figure 3. Hull definition nomenclature.

Values of residual drag coefficient ($C_{\Delta f}$) or roughness allowance (k_r) may be input to override the default values. Lateral loads are increased if there is a deck casing or keel. There is no reliable way to estimate these increases at present; the user must supply appropriate factors on the `Deck` and `Keel` records.

The following table summarizes the keyword records used to characterize hull components.

Keyword	Function
`BaseArea`	define hull base area
`Deck`	adjust sideforce and yawing moment for a deck casing
`Diameter`	diameter data block to follow
`HullSep`	force afterbody separation
`Keel`	adjust sideforce and yawing moment for a keel
`Label`	component label
`Length`	define component length
`LoD`	define length/diameter ratio
`MidHull`	define midpoint

`Nose`	define nose point
`Pitch`	define rotation in pitch
`Pop`	force termination of input for this component
`ResiDragR`	define hull residual drag coefficient or method of calculation
`Roll`	define rotation in roll
`RoughAllow`	define hull roughness allowance
`Source`	identify the data source for `HulFoM` load estimates
`Station`	station data block to follow
`Tail`	define tail point
`Yaw`	define rotation in yaw

A keyword record other than one of these will terminate input for the current component. After processing level 2 data is completed, the input stream is backspaced and the new record is re-read at level 1. The records are discussed below in terms of the functional group that they belong to.

5.1 Diameter and Station Records

These records, and the data blocks that follow, are alternative ways of defining hull section geometry.

`Diameter` $[N_{S_{in}}]$

Announces a data block of axial hull diameters at $N_{S_{in}}$ equally spaced stations ($N_{S_{in}} \leq 51$) from nose to tail. By default, $N_{S_{in}} = 21$. The data comprise three values of breadth, B, per record, apart from the last, which may have fewer values. The format is:

B_1 B_2 B_3
B_4 B_5 etc.

In generating the hull geometry, $T_i = B_i$ and $A_i = \pi B_i^2/4$, where T is height and A is cross section area.

If the number of stations for calculation, N_S, optionally specified on the `Hull` record, is not equal to $N_{S_{in}}$, the `Diameter` data will be interpolated. (Note that the `Hull` record must have been given both its arguments in order to specify N_S, e.g., '`Hull Default` N_S'.)

If the station data are not equally spaced, the `Station` record, which permits specification of the station axial coordinate, must be used.

`Station` $[N_{S_{in}}$, $[keyword]]$

Announces a block of section data for $N_{S_{in}}$ stations ($N_{S_{in}} \leq 51$). By default, $N_{S_{in}} = 21$. If the second optional argument is the keyword `Irregular`, the

stations are not equally spaced, and their axial coordinates must be provided in the data block. A value must be given for $N_{S_{in}}$ in order to specify irregular station spacing.

There is one record in the data block for each station, proceeding from nose to tail. If station spacing is irregular, the first value on each record is the station axial coordinate x_{in}. To simplify input this value is automatically scaled to hull length, so that input may run from 0 (nose) to 1 (tail), for example. The next (or first, for equal spacing) value is hull breadth B, followed by T (optional), A (optional), and z-wise camber F (also optional); all these values must be dimensional. Hull camber, by analogy with appendage section camber, is station mid-height offset from the nose–tail line, positive downwards. The example in annex D includes camber. If F is omitted, $F_i = 0$. If A is omitted, $A_i = \pi B_i T_i/4$, and if T is omitted, $T_i = B_i$ and $A_i = \pi B_i^2/4$.

If the stations are equally spaced, the `Station` data block format is:

B_1 [T_1 [A_1 [F_1]]]
B_2 [T_2 [A_2 [F_2]]]
etc.

to $N_{S_{in}}$. And if station spacing is irregular, it is:

x_{in_1} B_1 [T_1 [A_1 [F_1]]]
x_{in_2} B_2 [T_2 [A_2 [F_2]]]
etc.

to $N_{S_{in}}$. T_i, A_i, and F_i are enclosed in nested brackets to indicate that they are progressively optional from right to left.

5.2 Length, Midhull, Nose, Pitch, Roll, Tail, and Yaw Records

Hull component location and attitude, and therefore the definition of its local axis system, is determined by a combination of the keywords `Length`, `MidHull`, `Nose`, `Pitch`, `Roll`, `Tail`, and `Yaw`. The local x axis is directed forward and passes through the hull tailpoint, midpoint, and nosepoint. However, the hull may have camber in the direction of its z axis, in which case `MidHull` is simply the mean of `Nose` and `Tail`. To fully define hull location and attitude, the user must provide an unambiguous combination of these records. Redundant data are checked for consistency; inconsistency generates a fatal error. By default, the axis rotations, taken in order (Ψ, Θ, Φ), are assumed to be zero. Some examples are given at the end of this section.

`Length` L

Defines hull length between tailpoint and nosepoint. `Length` plus one of `Nose`, `MidHull`, or `Tail` provides an unambiguous hull axis definition for the default (zero) rotation. Rotation angles may be defined as required to override the default values.

`MidHull` x_m , $[y_m$, $z_m]$

Defines the midpoint of the hull component in vehicle coordinates; y_m and z_m are optional, and default to zero. `MidHull` plus one of `Nose` or `Tail` provides an unambiguous hull definition including yaw and pitch rotation angles; a `Roll` record is explicitly required to override the roll angle default. `MidHull` plus `Length` assumes the default (zero) rotation angles unless one or more of `Yaw`, `Pitch`, or `Roll` is also provided.

`Nose` x_n , $[y_n$, $z_n]$

Defines the nose point of the hull component in vehicle coordinates; y_n and z_n are optional, and default to zero. `Nose` plus one of `MidHull` or `Tail` provides an unambiguous hull definition including yaw and pitch rotation angles; a `Roll` record is explicitly required to override the roll angle default. `Nose` plus `Length` assumes the default (zero) rotation angles unless one or more of `Yaw`, `Pitch`, or `Roll` is also provided.

`Pitch` Θ

Defines pitch rotation of the local axis system with respect to the vehicle axis system, in degrees. Rotations are taken in the conventional order: yaw, pitch, roll. A value for Θ will be calculated if at least two records with the keyword `Nose`, `MidHull`, or `Tail` are present. Otherwise Θ will take the default value (zero) if a `Pitch` record is not provided.

`Roll` Φ

Defines roll rotation of the local axis system with respect to the vehicle axis system, in degrees. Rotations are taken in the conventional order: yaw, pitch, roll. If a `Roll` record is not provided, Φ will take the default value (zero).

`Tail` x_t , $[y_t$, $z_t]$

Defines the tail point of the hull component in vehicle coordinates; y_t and z_t are optional, and default to zero. `Tail` plus one of `Nose` or `MidHull` provides an unambiguous hull definition including yaw and pitch rotation angles; a `Roll` record is explicitly required to override the roll angle default. `Tail` plus `Length` assumes the default (zero) rotation angles unless one or more of `Yaw`, `Pitch`, or `Roll` is also provided.

`Yaw` Ψ

Defines yaw rotation of the local axis system with respect to the vehicle axis system, in degrees. Rotations are taken in the conventional order: yaw, pitch, roll. A value for Ψ will be calculated if at least two records with the keyword `Nose`, `MidHull`, or `Tail` are present. Otherwise, Ψ will take the default value (zero) if a `Yaw` record is not provided.

The following examples illustrate combinations of these keywords:

i. A hull component of length 10.829485 with midpoint located at $(-2.1461, -0.23341, 0.7705)$ in the vehicle axes. By default, local axes are aligned with vehicle axes.

```
MidHull  -2.1461  -.23341  .7705
Length   10.829485
```

ii. The same component given explicit rotations so that local axes are no longer aligned with the vehicle axes.

```
MidHull  -2.1461  -.23341   .7705
Length   10.829485
Yaw      -2.484513
Pitch    17.958312
Roll     27.5
```

iii. The previous component with two hull points defined, and therefore having implicit rotations in yaw and pitch. This and the previous example are exactly equivalent within truncation error.

```
Nose      3       -.4567   -.899
Tail     -7.2922  -.01012  2.44
Roll     27.5
```

iv. Again the same configuration, but with redundant information in pitch and yaw.

```
Nose      3       -.4567   -.899
Tail     -7.2922  -.01012  2.44
Yaw      -2.484513
Pitch    17.958312
Roll     27.5
```

Redundancy is overlooked; however, a small change or error in the data, such entering z_n as $-.889$ instead of $-.899$ in this example, will generate a fatal error because of inconsistency in input.

v. A hull component of length 5.3025 with midpoint located at (0, 0, 0), rotated implicitly by 90 degrees in pitch. Rotation in yaw defaults to zero.

```
Nose    0   0   -2.65125
Tail    0   0    2.65125
```

vi. The same result is obtained by explicitly defining the pitch angle.

```
MidHull  0  0  0
Length   5.3025
Pitch    90
```

5.3 BaseArea, Deck, HullSep, Keel, Label, LoD, ResiDragR, RoughAllow, and Source Records

These records provide additional hull parameters for the program.

`BaseArea` A_B

Defines a hull base area, which defaults to A_{N_S}, for the hull base drag calculation. It is required that $0 \leq A_B \leq \max A_i$.

`Deck` η_{Yd} , η_{Nd}

Defines load corrections for the presence of a deck casing. The hydrodynamic sideforce will be multiplied by η_{Yd} and the yawing moment by η_{Nd}. Default values are $(1.0, 1.0)$; i.e., there is no deck casing. While there are no systematic data from which to select these corrections, reference [12] illustrates a case for which values of $(1.2, 1.0)$ may be appropriate. The user may be able to deduce suitable values by tuning predictions with trials data.

`HullSep` N_{sep}

This record forces hull separation to occur at hull station N_{sep}, where N_{sep} is the mandatory integer argument. A large value of the argument, say 99, will set N_{sep} equal to the last station, N_S.

Specifying N_{sep} allows the user to address the '`No minimum found in dA/dx (DT2) for separation`' error (section 3.3), or to override the estimated location of separation, which is output on the *Root*`.log` file.

`Keel` η_{Yk} , η_{Nk}

Defines load corrections for the presence of a keel. The hydrodynamic sideforce will be multiplied by η_{Yk} and the yawing moment by η_{Nk}. Default values are $(1.0, 1.0)$; i.e., there is no keel. While there are no systematic data from which to select these corrections, reference [12] illustrates a case for which values of $(1.7, 0.9)$ may be appropriate. The user may be able to deduce suitable values by tuning predictions with trials data.

`Label` *label string*

This record provides a user-defined component label. The label string consists of '`#`' or '`?`' followed by up to 24 alphabetic characters; case is ignored. If there is no `Label` record, the default label is `#HULL`*nnn* where *nnn* is the number of this hull component as encountered in the input file (i.e., *nnn* = 001 for the first, 002 for the second, etc.).

`LoD` L/D_M

Overrides the calculated value of length/diameter ratio for the current hull component. The argument is mandatory, and has a permitted range of $4 \leq L/D_M \leq 16$.

`ResiDragR` $C_{\Delta f}$ or *keyword*

Defines a hull residual drag coefficient $C_{\Delta f}$ or the method for its estimation. The two methods available are the semiempirical expression given by Hoerner [17] for bodies of rotation (*keyword* = `Hoerner`), or a fit to data for torpedo-like bodies (*keyword* = `Torpedo`). If the numerical argument is given, then it is required that $0 \leq C_{\Delta f} \leq 0.5$. The default for hull residual drag is calculation by Hoerner's method.

`RoughAllow` k_r

Defines a coefficient k_r for the hull roughness allowance in hull drag. By default, $k_r = 0.0004$; the user-input value must be $0 \leq k_r \leq 0.001$.

`Source` *keyword*

The `HulFoM` (default) hull load calculations use lookup tables derived from model test data. There are two sets of tables in the present build, one derived from Kriging interpolation, the other from neural network generalization; they are discussed in reference [12]. The keyword argument indicates which to select: `Kriging` or `NeuralNet`. The default is `Kriging`.

5.4 Pop

The `Pop` record forces termination of level 2 input, completing processing of the current component and returning to level 1 (an instance in which the order of input records is important). Normally, an unrecognized keyword (or the end of file) triggers this sequence, but the proliferation of keywords with program development may make this difficult in future.

6 Root.geo Level 2 Lift Input

The nomenclature used to define basic lifting appendage geometry is illustrated in figure 4. The four corner points $c1, c2, c3$, and $c4$ must be given in vehicle axes. Appendage sections are assumed to be NACA 4–digit, and thickness/chord ratio and other parameters are optional. An isolated appendage is assumed to be mounted on a reflecting plane at the root, i.e., loading is finite at the root because of lift carry-over. An unattached symmetrical wing would therefore be modeled by two isolated appendages, one for each semispan. All other appendage configurations require a type specification (`Rudder`, `Sail`, etc.) in order to model interference effects.

Figures 5, 6, and 7 illustrate a number of the additional geometric parameters needed to account for interference; they are defined in the nomenclature. The general approach to interference is modification of the local angle of attack so that the appendage normal (Z) and axial (X) forces, in nondimensional coefficients, are of the form:

$$
\begin{aligned}
C_Z &= C_{Z\alpha}\,\alpha_{(1)} + C_{Z\alpha\alpha\alpha}\,\alpha^3_{(2)} \\
C_X &= C_{X0} + C_{X\alpha\alpha}\,\alpha^2_{(3)} + \ldots
\end{aligned} \tag{3}
$$

where $\alpha_{(1)}$, $\alpha_{(2)}$, and $\alpha_{(3)}$ are equivalent angles of attack that are functions of local geometric angles of attack α, and deflection δ, and of the interference parameters. That is,

$$
\begin{aligned}
\alpha_{(1)} &= \alpha_{(1)}\,(\alpha, \delta, r, b_{exp}, b_1, b_2, \ldots) \\
&\text{etc.}
\end{aligned} \tag{4}
$$

A number of non-geometric parameters may be defined where it is necessary to override default values assigned by the code. These include coefficients for junction drag (C_{Xj}) and zero lift drag (C_{X0}), a flap gap correction (η), flap profile drag (k_δ), and tailplane efficiency (K_{WB}, k_{WB}).

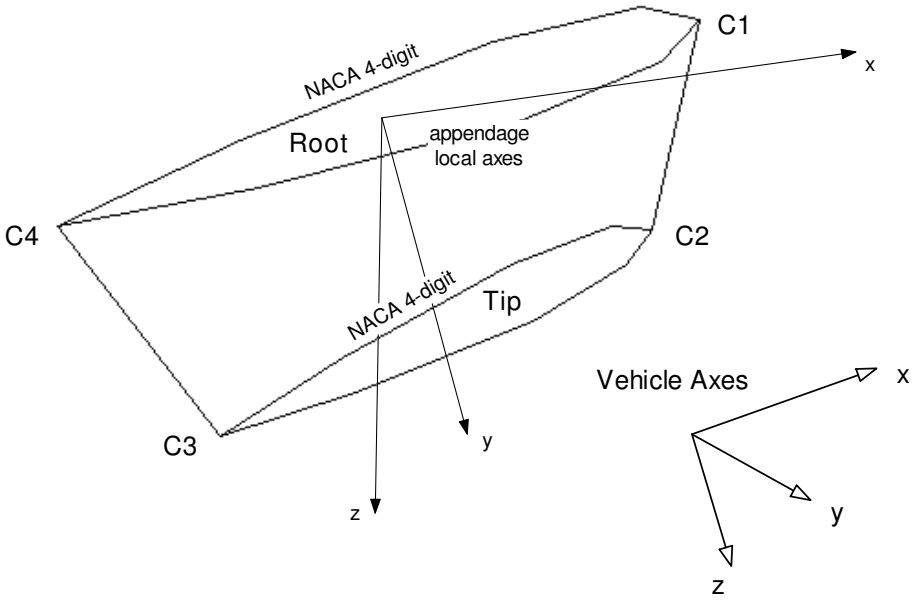


Figure 4. Appendage definition nomenclature.

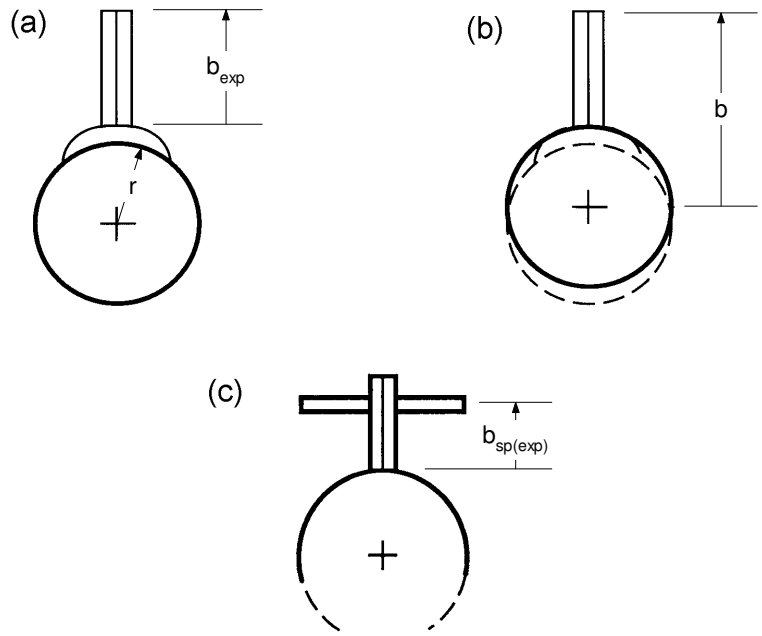


Figure 5. Hull/sail input nomenclature: (a) non-axisymmetric hull, (b) equivalent axisymmetric hull, and (c) sailplane location.

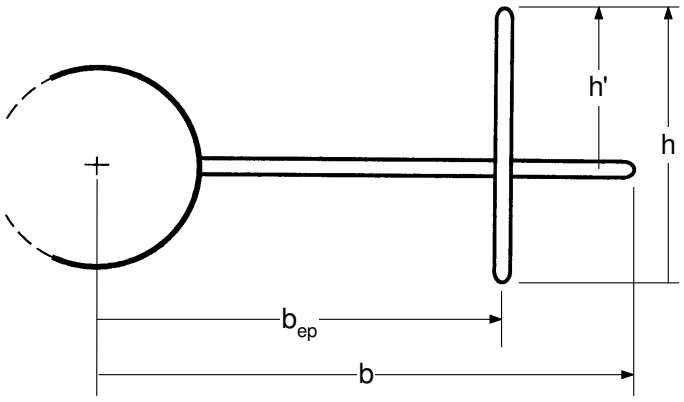


Figure 6. Endplate input nomenclature.

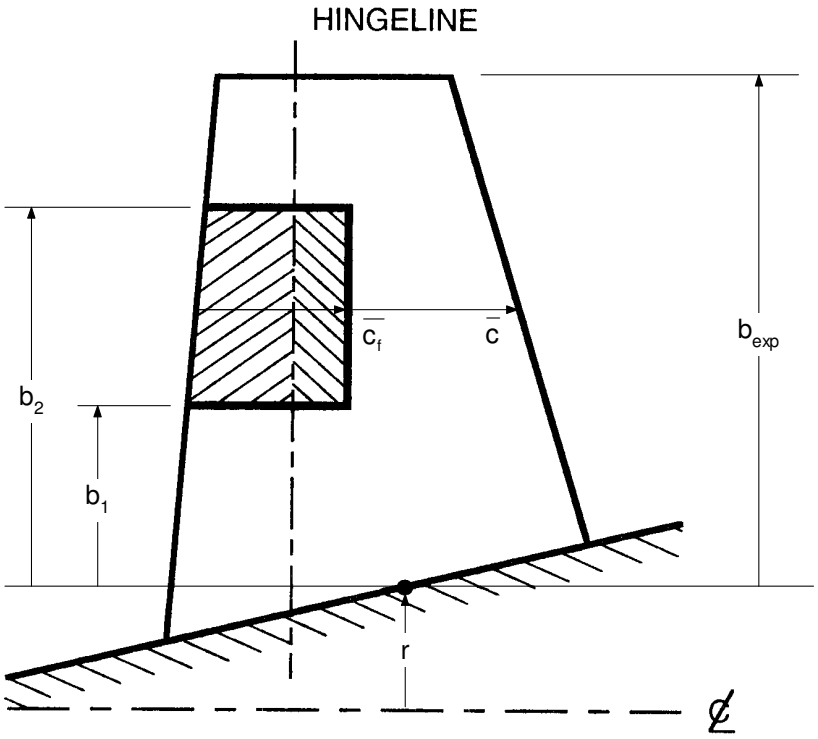


Figure 7. Flapped appendage input nomenclature.

The following table summarizes the keyword records used to characterize lifting appendages:

Keyword	Function
`AllMoving`	define control type as All-Moving
`Camber`	apply geometric camber
`CXJ`	define junction axial force coefficient
`CXZero`	define zero lift axial force coefficient
`DelNonLin`	enable nonlinear flap calculation
`DelRateLim`	set control deflection rate limit
`Del`	set initial control deflection
`Delta`	set initial control deflection
`DeltaDyn`	set control dynamic parameters
`DeltaLim`	set control deflection limits
`FlapGap`	define flap balance and gap correction
`Flapped`	define control type as Flapped

`FlaProDrag`	define flap profile drag
`HasEndP`	correct for having an endplate on this appendage
`IsEndP`	correct for this appendage being an endplate
`Kill`	save for a copy, but do not use, this appendage
`Label`	component label
`LBowplane`	define appendage type as Large Bowplane
`Planform`	appendage corner data block follows
`Pop`	force termination of input — see section 5.4
`Reflected`	copy saved appendage by reflecting
`Rotated`	copy saved appendage by rotating
`Rudder`	define appendage type as Rudder
`Sail`	define appendage type as Sail
`Sailplane`	define appendage type as Sailplane
`Save`	save this appendage for subsequent copy
`SBowplane`	define appendage type as Small Bowplane
`ShipStab`	define appendage type as Ship Stabilizer
`Sternplane`	define appendage type as Sternplane
`Tail`	define appendage type as Tail
`TailEff`	define tail efficiency
`ToC`	define section thickness/chord ratio
`Twist`	apply geometric twist
`WeightDCI`	set indirect control deflection weights

A keyword record other than one of these will terminate input for the current component. After processing of level 2 data is completed, the input stream is backspaced and the new record is re-read at level 1. The records are described below in terms of the functional group that they belong to.

6.1 The Planform Record

The appendage planform is described by four corner points as illustrated in figure 4. For a triangular appendage, points $c2$ and $c3$ should be made coincident.

`Planform`

The next four records contain coordinates of the four appendage corner points in vehicle axes. The corner points are input one point per record, and must be in order: root leading edge, tip leading edge, tip trailing edge, and root trailing edge. The planform data block format is therefore:

$$\begin{array}{lll} x_{c1} & y_{c1} & z_{c1} \\ x_{c2} & y_{c2} & z_{c2} \\ x_{c3} & y_{c3} & z_{c3} \\ x_{c4} & y_{c4} & z_{c4} \end{array}$$

6.2 Type Records: LBowplane, Rudder, Sail, Sailplane, SBowplane, ShipStab, Sternplane, and Tail

The keywords which define appendage type are `LBowplane`, `Rudder`, `Sail`, `Sailplane`, `SBowplane`, `ShipStab`, `Sternplane`, and `Tail`. Defining the appendage type ensures that appropriate interference corrections are made to the appendage forces, and may have additional implications. For example, although interference corrections are identical for types `Rudder`, `Sternplane`, and `Tail`, these types respond differently to control commands. Only one appendage type record may be supplied for each lifting component; if no type is specified, it is treated as an isolated appendage.

Appendage type records have at least one argument: a parameter r that is used to estimate lift carry-over. For the general case of an appendage mounted on a hull, and where local dihedral is minimal (i.e., the appendage is more-or-less normal to the hull surface), r is the hull radius at the appendage mid root.

`LBowplane` r

The appendage is defined as a large bowplane. Whether a bowplane is considered large or small depends on the ratio of its dimensions to the local hull radius [4]. A small bowplane is corrected for dihedral (below), a large one is not. Argument r is the mid-root hull radius; $r \geq 0$.

`Rudder` r, $[r_M]$

The appendage is defined as a rudder. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument r_M is maximum hull radius, required for Dempsey's calculation of tailplane efficiency [18]; by default $r_M = r$. Tailplane efficiency is specified with a `TailEff` record.

`Sail` r

The appendage is defined as a sail. Argument r is the mid-root hull radius; $r \geq 0$.

`Sailplane` r, $b_{sp(exp)}/b_{exp}$

The appendage is defined as a sailplane; both arguments are mandatory. Argument r is the mid-root to sail centerline distance (about half the sail thickness); $r \geq 0$. Parameters $b_{sp(exp)}$ and b_{exp} are exposed sailplane height (i.e, above the sail root) and exposed sail span, see figure 5. The permitted range of the argument is $0.0 \leq b_{sp(exp)}/b_{exp} \leq 1.0$.

`SBowplane` r, $[\phi]$

The appendage is defined as a small bowplane. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument ϕ is the dihedral or anhedral angle (sign is immaterial) with respect to the hull tangent, in degrees. The default value of ϕ is $0°$, and its permitted range is $-90° \leq \phi \leq 90°$.

`ShipStab` r, $[x']$

The appendage is defined as a ship stabilizer. Argument r is the local hull radius; $r \geq 0$. Optional argument x' is downstream distance between the origin of the hull boundary layer (typically, near the forwardmost point of the hull) and the origin of the appendage axis system, in metres, and must not be negative; x' defaults to zero.

`Sternplane` r, $[r_M]$

The appendage is defined as a sternplane. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument r_M is maximum hull radius, required for Dempsey's calculation of tailplane efficiency [18]; by default $r_M = r$. Tailplane efficiency may be specified with a `TailEff` record.

`Tail` r, $[r_M]$

The appendage is defined to be a tail fin. Argument r is the mid-root hull radius; $r \geq 0$. Optional argument r_M is maximum hull radius, required for Dempsey's calculation of tailplane efficiency [18]; by default $r_M = r$. Tailplane efficiency may be specified with a `TailEff` record.

6.3 Control Type Records: AllMoving and Flapped

The records which define control type are `AllMoving` and `Flapped`.

`AllMoving` $[S_m/S_{exp}]$

The appendage is an all-moving control surface; the argument is optional. Parameters S_m and S_{exp} are area of the moving part and total exposed area of the appendage. The default value of S_m/S_{exp} is 1.0, and its permitted range is $0.1 < S_m/S_{exp} \leq 1.0$.

`Flapped` $[b_1/b_{exp}\ ,\ b_2/b_{exp}]\ ,\ [\overline{c_f}/\overline{c}]$

The appendage is a control surface with a trailing edge flap. The arguments are the inner and outer spanwise flap fractions, and the chordwise flap fraction, see figure 7.

Note that $\overline{c}$ is sometimes defined from mid trailing edge to mid leading edge, and $\overline{c_f}$ is aligned accordingly; however, it is preferable to make them normal to the flap hinge line as shown in the figure.

The first two arguments of `Flapped` are optional as a pair; the third argument is also optional. Thus, if only one argument is present, it is assumed to be $\overline{c_f}/\overline{c}$; if two are present, they are assumed to be b_1/b_{exp} and b_2/b_{exp}. Default values for the arguments are 0.0, 1.0, and 0.25 respectively, and their permitted ranges are $0.0 \leq b_1/b_{exp} \leq 1.0$, $b_1/b_{exp} \leq b_2/b_{exp} \leq 1.0$, and $0.0 \leq \overline{c_f}/\overline{c} \leq 1.0$.

6.4 Control Parameter Records: DelNonLin, Del, Delta, DeltaLim, FlapGap, FlaProDrag, and WeightDCI

These records set or redefine additional parameters for a control appendage. `Del` is a synonym for `Delta`; making it a keyword allows the common abbreviation `Del` to be used without ambiguity.

`DelNonLin`

This record enables the calculation of effective flap angle, $\delta_e < \delta$, for an appendage with a trailing edge flap [4]. The keyword record `Flapped` must also appear in the appendage specification. Without the `DelNonLin` record, $\delta_e = \delta$.

`Del` $\delta_\bullet$

Defines an initial value, $\delta_\bullet$, in degrees, for the control deflection; the default is zero. The initial deflection may have the effect of offsetting the reference for some control commands. It can be redefined at several places in the *Root*`.sim` file. This record applies to both `AllMoving` and `Flapped` control types.

`Delta` $\delta_\bullet$

As `Del`, above.

`DeltaLim` δ_{Max} , δ_{Min}

Sets control deflection limits: $\delta_{Max} \geq \delta \geq \delta_{Min}$. The defaults are $+35°$ and $-35°$. These limits can be redefined at several places in the *Root*`.sim` file. The record applies to both `AllMoving` and `Flapped` control types.

`FlapGap` η

This record overrides the default value of 1.0 for the flap balance and gap efficiency correction, η. It is valid only for an appendage with a trailing edge flap — the keyword record `Flapped` must also appear in the appendage specification. The argument is mandatory and has a permitted range of $0.8 \leq \eta \leq 1.25$.

`FlaProDrag` k_δ

This record overrides the default value of -0.5 for the coefficient of profile axial force (i.e., negative drag) associated with a trailing-edge flap deflection, k_δ (which is nondimensionalized with respect to exposed appendage area). It is valid only for an appendage with a trailing edge flap — the keyword record `Flapped` must also appear in the appendage specification. The argument is mandatory and has a permitted range of $-2.0 \leq k_\delta \leq 0.0$.

`WeightDCI` W_{ID} , W_{IH} , $[W_{IR}]$

It will be seen in section 14 that there is an indirect method of vehicle control in which an effective deflection angle is associated with all appendages capa-

ble of controlling the vehicle in a particular mode (depth, heading, or roll, using subcommands like `Up`, `Starboard`, etc.). The indirect method provides the mechanism for autopilot control of the vehicle. Appendages are assigned weights that determine the degree to which they respond in each of these indirect control modes.

Mandatory arguments W_{ID} and W_{IH}, and optional argument W_{IR}, are nondimensional weights for indirect depth, indirect heading, and indirect roll control respectively. The weights multiply a commanded deflection and can be used, for example, to balance bowplane and sternplane response to depth control commands, or (with a value of zero) to decouple a particular appendage from an indirect control mode. These weights must be in the range $-2 \leq W_{ID,H,R} \leq 5$, and are by default equal to 1.0. They can be redefined at several places in the *Root*`.sim` file. This record applies to both `AllMoving` and `Flapped` control types.

6.5 Control Dynamics Records: DelRateLim and DeltaDyn

Two records are used to specify the dynamics of a control deflection: `DelRateLim` and `DeltaDyn`. They apply to both `AllMoving` and `Flapped` control types.

`DelRateLim` $\dot{\delta}_M$

Argument $\dot{\delta}_M$ is the maximum control deflection rate in degree/s; it must be positive. The default value is 9 degree/s.

`DeltaDyn` ζ_δ , ω_δ

Argument ζ_δ is the nondimensional control deflection damping and ω_δ is the control response frequency in rad/s. These parameters, together with $\dot{\delta}_M$, determine control deflection time histories by way of a second order ODE for each control appendage. Damping ζ_δ must be positive, and a warning is given if it is > 1.0; the default value is 0.9. Frequency ω_δ must be positive; its default value is $20\,\dot{\delta}_M/(\delta_{Max} - \delta_{Min})$, which is equivalent to 10 $\times$ maximum rate / |maximum deflection|.

6.6 Endplate Records: HasEndP and IsEndP

The keywords which control endplate specifications are `HasEndP` and `IsEndP`.

`HasEndP` h/b , $[b_{ep}/b]$

The current appendage has an endplate; the first argument is mandatory, the second is optional. Parameters h, b, and b_{ep} are endplate height, appendage semispan, and spanwise location of the endplate (the last two measured from the hull centerline if appropriate), see figure 6. Permitted range of the first

argument is $0.0 \leq h/b \leq 2.5$. The default value of the second argument is 1.0 and its permitted range is $0.0 \leq b_{ep}/b \leq 1.0$.

`IsEndP` $[h'/h]$

The current appendage is an endplate; the argument is optional. Parameters h' and h are point of attachment relative to the full span and the full span ($h = 2b$), or height. The default value of the argument is 0.5, and its permitted range is $0.0 \leq h'/h \leq 1.0$.

6.7 Save Records: Kill and Save

An appendage configuration can be saved, with or without itself being used, for re-use through reflection or rotation. The keywords which control saving are `Kill` and `Save`; both overwrite a previous save. Note that the component label is not part of the configuration; it is not saved, while all other attributes are.

`Kill`

The current appendage configuration will be saved for a subsequent copy, but will not itself be used. The user thus does not have to calculate corner points for a non-vertical or non-horizontal appendage that is more easily defined in the vehicle xy or xz plane and then rotated into its correct orientation.

`Save`

The current appendage configuration will be used and then saved for a subsequent copy.

6.8 Copy Records: Reflected and Rotated

The keywords which control copying the most recently saved appendage are `Reflected` and `Rotated`. Only planform corner points are modified by a copy; all other physical attributes of the appendage remain the same. The label is not transferred, so that a `Label` record is required for the new appendage or the default will be used.

These keywords cannot be used in combination with any others in level 2 input except for a `Label` record.

`Reflected`

The saved appendage is reflected in the vehicle xz plane. This operation is illustrated for a sailplane in annex D.

`Rotated` ϕ_R

The saved appendage is rotated ϕ_R degrees about the vehicle x axis. The argument is mandatory; ϕ_R is positive rotating from the y axis to the z axis,

and there is no restriction on its value. This operation is illustrated for tail appendages in annex D.

6.9 Camber, CXJ, CXZero, Label, TailEff, ToC, and Twist Records

Keywords `Camber`, `CXJ`, `CXZero`, `Label`, `TailEff`, `ToC`, and `Twist` override default values of various lifting surface parameters.

`Camber` f

This record overrides the default value of 0.0 for nondimensional geometric section camber, f. The argument is mandatory, and is positive in the direction of increasing z in the local appendage axis system.

`CXJ` C_{Xj}

This record overrides the default value of 0.0 for the junction axial force coefficient, C_{Xj}, i.e., 'junction drag'. The argument is mandatory and must not be positive.

`CXZero` C_{X0}

This record overrides the default value of -0.01 for the zero lift axial force coefficient, C_{X0} (which is nondimensionalized with respect to appendage area). The argument is mandatory and must not be positive.

`Label` *label string*

This record provides a user-defined component label consisting of '`#`' or '`?`' followed by up to 24 alphabetic characters; case is ignored. If there is no `Label` record, the default label is `#LIFT`*nnn* where *nnn* is the number of this lifting appendage as encountered in the input file.

`TailEff` *keyword* or K_{WB} , $[k_{WB}]$

This record controls estimation of the tailplane efficiency associated with a vehicle at incidence, K_{WB}, and the tailplane efficiency associated with a control deflection, k_{WB}. The argument is a keyword, `Bohlmann` or `Dempsey`, or one or two numerical values.

For all tailplane appendages (`Rudder`/`Sternplane`/`Tail`), Bohlmann-Spreiter calculations [19] are done for inviscid flow and the resulting efficiency ratio k_{WB}/K_{WB} is saved.

If the keyword argument is `Bohlmann`, or if the `TailEff` record is absent altogether, the efficiency ratio is ignored and viscous Bohlmann-Spreiter calculations are done for each evaluation of tailplane efficiency in a simulation. Additional corrections are made for the influence of tail cone angle and displacement thickness at small angles of incidence as discussed in reference [19].

If the keyword argument is `Dempsey`, K_{WB} is calculated by Dempsey's method [18] and k_{WB} is derived using the inviscid efficiency ratio; these values are then constant with respect to speed.

For numerical input, the first value is K_{WB}. The second value, if present, is k_{WB}, otherwise k_{WB} is derived using the inviscid efficiency ratio; in either case these values are constant for all speeds. The limits for numerical input are $0 \leq K_{WB} \leq 1.5$ and $0 \leq k_{WB} \leq K_{WB}$.

`ToC` t/c

This record overrides the default value of 0.12 for section thickness/chord ratio, t/c. The argument is mandatory and has a permitted range of $0.0 \leq t/c \leq 0.25$.

`Twist` α_b

This record overrides the default value of 0.0 for geometric section twist, α_b, defined as the angle of attack at the tip relative to the root. The argument is mandatory; its value is in degrees and has a permitted range of $-30^\circ \leq \alpha_b \leq 30^\circ$. Twist is positive in the sense of rotating the appendage local z axis into the x axis. This record is ignored, with a warning, for triangular planforms and for planforms that include twist in the `Planform` specification.

7 Root.geo Level 2 OOPCalc Input

Background to the out-of-plane load calculations implemented in `DSSP21` is given in reference [20]. These loads occur on a hull with asymmetric appendages attached, the most common case being the out-of-plane normal force and pitching moment due to the sail on a submarine. In `DSSP21`, out-of-plane loads are calculated by default for each appendage with the type `Sail`, using the nearest hull if there is more than one. The `OOPCalc` record and associated level 2 input are used to override default parameters, to specify calculations for other than the default condition, or to suppress the out-of-plane calculations altogether. To suppress them, use '`OOPCalc None`'; furthermore, any use of an `OOPCalc` record suppresses all the default calculations.

For dynamic (captive and free) simulations, the out-of-plane calculations create a time history by convecting the load distribution aft at the instantaneous value of u. If u becomes large, there may be too few points defining the distribution to maintain precision in the load estimates, and a warning is issued:

```
?? Warning from OOPSTEP
   Depth of OOP stack less than four data points
   - precision lost in load integration
```

The user must decide whether accurate out-of-plane loads are significant for the manoeuver under consideration, and reduce the step size accordingly. However, it is gen-

erally safe to ignore this warning over short periods, especially if vehicle depth is under autopilot control.

The following table summarizes the keyword records used in `OOPCalc` level 2 input:

Keyword	Function
`Hull`	identify the hull component
`KParam`	define nonlinearity of the load distribution
`Lambda`	define the extent of the hull bound vortex
`Lift`	identify the lift component
`Pop`	force termination of input — see section 5.4

Of these records, only `Lift` is always mandatory.

7.1 Hull, KParam, Lambda, and Lift Records

`Hull` *label string* or N_H

The argument identifies the hull to be used in the calculation, with a label string or integer N_H. If this record is omitted, the nearest hull is used.

`KParam` k

The form of the out-of-plane load distribution suggested in reference [20] consists of a linear ramp function raised to the power m, where $m = 1 + k|\beta_c|$, to account for nonlinearity. β_c is the sail angle of incidence in radians. Reference [20] illustrates a number of submarine configurations for which $-2 \leq k \leq 3$. By default, $k = 0$, representing a linear load distribution.

`Lambda` λ_1 , λ_2

The arguments define the extent of the hull-bound vortex over which the out-of-plane load is distributed. At the forwardmost extent, $\lambda_1 = 0$, which corresponds to the sail trailing edge, and at the aftmost, $\lambda_2 = 1$, which corresponds to the hull AP. It is required that $0 \leq \lambda_1 < \lambda_2 \leq 1$. By default, $\lambda_1 = 0$ and $\lambda_2 = 1$. It is concluded in reference [20] that there is at present little data to justify changing these values.

`Lift` *label string* or N_L

The argument identifies the lifting appendage (which may be of any type) to be used in the calculation, with a label string or integer N_L.

8 Root.geo Level 2 Propulsor Input

Propulsor types are discussed above, in section 4.4. The input discussed in this section is applicable to all propulsor types: `DSSPtwo`, `STRprop`, and `WageningB`.

The following table summarizes the keyword records used to characterize a generalized propulsor:

Keyword	Function
`Diameter`	define propulsor diameter
`Label`	propulsor label
`Left`	left-handed propulsor
`Location`	define propulsor local origin
`Pitch`	define local axis pitch
`Pop`	force termination of input — see section 5.4
`Reflected`	copy saved propulsor by reflecting
`RevDynam`	set RPM dynamic parameters
`RevLimit`	set maximum RPM
`RevRateLim`	set RPM change rate limit
`Right`	right-handed propulsor
`Save`	save this propulsor for subsequent copy
`Yaw`	define local axis yaw

The `Label` definition is analogous to that of other components. If there is no `Label` record, the default label is `#PROP`*nnn* where *nnn* is the number of this propulsor as encountered in the input file.

8.1 Propulsor Geometry Records: Diameter, Location, Pitch, and Yaw

`Diameter` is self-explanatory; keywords `Location`, `Pitch`, and `Yaw` define, via the propulsor's location and orientation, the origin and orientation of its thrust and torque vectors. The distinction between a propulsor's physical location and the points of application of its forces and moments may have to be considered.

`Diameter` D_P

This record defines propulsor diameter D_P in metres; it is mandatory.

`Location` x_P , $[y_P$, $z_P]$

The propulsor local origin, (x_P, y_P, z_P) in vehicle axes, is the point at which estimated thrust and torque are applied; y_P and z_P may be omitted, in which case they both default to zero. For most purposes, the location of the propulsor hub can be used. A `Location` record must be present.

`Pitch` θ_P

θ_P defines the pitch of the local propulsor axes with respect to vehicle axes, in degrees. The order of axis rotation is ψ_P, θ_P. If this record is omitted, θ_P defaults to zero.

`Yaw` ψ_P

ψ_P defines the yaw of the local propulsor axes with respect to vehicle axes, in degrees. The order of axis rotation is ψ_P, θ_P. If this record is omitted, ψ_P defaults to zero.

8.2 Propulsor Parameter Records: Left, RevLimit, and Right

`Left`

The propulsor is left-handed. For self-propulsion ahead, the propulsor has counterclockwise rotation looking forward, so that torque on the vehicle is positive. Right-handedness is the default.

`RevLimit` n_M

Argument n_M is maximum RPM; it must be positive. The default value is $15.433/D_P$ sec^{-1}, i.e., RPM for $J_S = 1$ at 30 kt.

`Right`

The propulsor is right-handed. For self-propulsion ahead, the propulsor has clockwise rotation looking forward, so that torque on the vehicle is negative. This record is redundant since right-handedness is the default.

8.3 Propulsor Dynamics Records: RevDynam, and RevRateLim

`RevDynam` ζ_n , ω_n

Argument ζ_n is the nondimensional change-of-RPM damping and ω_n is the change-of-RPM response frequency in rad/s. These parameters, together with $\dot{n}_M$, determine RPM time histories by way of a second order ODE for each propulsor. Damping ζ_n must be positive, and a warning is given if it is > 1.0; the default value is 0.9. Frequency ω_n must be positive; its default value is $10\,\dot{n}_M/n_M$ where n_M is maximum RPM.

`RevRateLim` $\dot{n}_M$

Argument $\dot{n}_M$ is the maximum change-of-RPM rate in RPM/s; it must be positive. The default value is 10 RPM/s.

8.4 Save/Copy Records: Reflected, and Save

Similarly to lifting appendages (sections 6.7 and 6.8), a propulsor may be saved and copied; however, only `Save` and `Reflected` records are applicable. Note that the propulsor label is not saved, while all other attributes are.

`Reflected`

The saved propulsor is reflected in the vehicle xz plane. A `Label` record is required for the new propulsor or the default `#PROP`nnn will be used.

`Save`

The current propulsor will be used and then saved for a subsequent copy. This overwrites a previous save.

9 Root.geo Level 2 WagBInit Input

Input discussed in this section is used to initialize propulsors with type `WageningB` (section 4.4): i.e., Wageningen B4–70 series propellers [15]. If this type is used, all propulsors must be the same type. The initialization estimates optimum J_S and RPM/kt values for given operational conditions.

Wageningen B4–70 propeller thrust and torque characteristics are derived from lookup tables taken from table 7 in reference [15]. It is assumed that all propellers contribute to forward thrust; for initialization, the total thrust is apportioned between them in a specified ratio. The same form of calculation is used for all, so the program does not accurately handle the situation of a main propulsor of this type on the hull axis with an auxiliary propulsor on each sternplane (i.e., effectively in open water).

The following table summarizes the keyword records for level 2 `WagBInit` input.

Keyword	Function
`Diameter`	define a hull reference diameter
`Open`	use open water propeller option
`PoD`	set pitch/diameter ratio
`Pop`	force termination of input — see section 5.4
`Rho`	set fluid density for propeller initialization
`TRatio`	set thrust ratios for multiple propellers
`Ukt`	ahead speed for propeller initialization, in kt
`Ums`	ahead speed for propeller initialization, in m/s

9.1 Diameter, Open, PoD, Rho, TRatio, Ukt, and Ums Records

`Diameter` D_M

The argument, D_M, is the maximum diameter of the hull that is used to form the ratio D_P/D_M for estimating wake fraction and thrust deduction coefficients. If the vehicle has only one hull, D_M defaults to its calculated maximum diameter; if there is none, or more than one hull, a value for D_M must be input explicitly.

`Open`

Forces the open water condition of wake fraction and thrust deduction equal to zero, and may be appropriate for using `WageningB` propellers in the old twin-screw submarine arrangement. The default is to estimate wake fraction and thrust deduction for propellers in the wake of a coaxial hull.

`PoD` P_P/D_P

This record overrides the default estimate of optimum pitch/diameter ratio for `WageningB` propellers. It should be within the range $0.6 \geq P_P/D_P \geq 1.4$. If it is input or calculated in the range $0.5 \geq P_P/D_P \geq 0.6$, it is simply reset to 0.6. If $P_P/D_P < 0.5$ or $1.4 < P_P/D_P$, the user is given the option to continue using the limiting value 0.6 or 1.4 respectively, otherwise the program is terminated.

`Rho` ρ

This record overrides the fluid density default value of 1000 kg/m^3 for `WageningB` propeller initialization (i.e., it does not permanently reset ρ). The argument should be in the range $990 \leq \rho \leq 1035$.

`TRatio` $\widetilde{T}_i$, $i = 1, \ldots$

For initializing multiple `WageningB` propellers, the ratio, $\widetilde{T}_i$, of thrust developed by each one to the total thrust can be specified. There must be one value of $\widetilde{T}_i$ for each propeller. They will be normalized by their total, $\sum_i \widetilde{T}_i$, and so can be in a convenient form such as estimated thrust values or percentages. If this record is absent, $\widetilde{T}_i$ defaults to the disk area of the i^{th} propeller, so that each one is nominally equally loaded.

`Ukt` $U_\bullet$ (kt)

This record overrides the default value of 5 kt for vehicle speed used to initialize J_S and RPM/kt for `WageningB` propellers. There can be only one `Ukt` or `Ums` record.

`Ums` $U_\bullet$ (m/s)

This record overrides the default value of 2.57222 m/s (= 5 kt) for vehicle speed used to initialize J_S and RPM/kt for `WageningB` propellers. There can be only one `Ums` or `Ukt` record.

10 Root.geo Level 2 Autopilot Input

Before listing level 2 autopilot input, a brief outline of the control algorithms will be useful.

The error signal J_E, for a quantity Q (i.e., heading, roll, etc) is in general:

$$J_E = \left[Q_c - Q\right] C_E \tag{5}$$

where Q_c is the commanded value of Q, and C_E (in units of Q^{-1}) is an error gain required to normalize J_E to within $-1 \geq J_E \geq 1$ for a typical maximum response. Such definitions are deliberately vague; some trial and error will be required to find optimum values for the gains and other autopilot parameters.

For propulsion control, the error signal has the form $\left[Q_c^n - Q^n\right] C_E$, where $n = 1/3$ if Q is power, $n = 1$ if Q is speed, or $n = 1/2$ if Q is thrust, and the units of C_E correspond in each case.

The depth error signal is also a little more complex than equation (5) as it depends on pitch angle, θ, in addition to depth, z_0:

$$J_{E\,(\text{depth})} = \left[\left[z_{0c} - z_0\right]_{-l_z \sin\theta_L}^{+l_z \sin\theta_L} \; + \; l_z \sin\theta\right] C_{E\,(\text{depth})} \tag{6}$$

where l_z is the lookahead distance (making it appear that depth is sensed l_z ahead of the coordinate origin), and θ_L is the autopilot pitch limit. In formulating $J_{E\,(\text{depth})}$, the full equation (6) is used by default, although use of the depth or pitch contributions alone can be invoked during a manoeuvering simulation, section 22.4.

In all cases, J_E is clipped at ± 1. There are two methods in `DSSP21` for deriving the corresponding control signal J_C. The first is a phase lead algorithm:

$$\dot{J}_C = \frac{\alpha}{\tau}\left[J_E + \tau \dot{J}_E - J_C\right] \tag{7}$$

where α is the amplitude gain of the transformation and τ is the anticipation time constant [21]. After rearrangement, equation (7) is solved as a first order ODE in the time integration.

The second method for deriving J_C is a PID (Proportional-Integral-Derivative) algorithm:

$$J_C = K_P\, J_E + K_I \int J_E\, dt + K_D\, \dot{J}_E \tag{8}$$

where K_P, K_I, K_D are the PID coefficients. To solve this in the time integration, a lightly damped error signal J'_E is used in the derivative term.

$$k_1\, \dot{J}'_E + k_2\, J'_E = J_E \tag{9}$$

The control signal is then obtained from two first order ODEs using both the damping equation coefficients, k_1 and k_2, and the PID coefficients.

J_C is clipped at ± 1 and multiplied by a control gain C_C in appropriate units (degrees for control deflections, RPM for propulsion revolutions) to determine the control correction. Time histories of J_C and J_E for the various autopilots are tabulated in the auxiliary output file `aux`*nnn*`.txt`.

Defaults are provided for all the parameters discussed above, but optimizing them will require some trial and error depending on the geometric and other characteristics of the vehicle. None of the records listed in sections 10.1 to 10.4 is mandatory. However, if a record is input, all the arguments for it must be included.

10.1 AutoDepth Level 2 Records: DepConGain, DepErrGain, LookAhead, PhaseLead, PID, and PitchLimit

`DepConGain` $C_{C\,(\text{depth})}$

Depth control gain $C_{C\,(\text{depth})}$ is in degrees, and should be positive. Since the commanded deflection depends on appendage dihedral and control weight W_{ID} (sections 6.4 and 14.3), this parameter can exceed the maximum deflection on an appendage. Its default value is 20 degrees.

`DepErrGain` $C_{E\,(\text{depth})}$

Depth error gain $C_{C\,(\text{depth})}$ is in m^{-1}, and should be positive. Based on a lookahead of $\ell/2$ and the default pitch limit (10 degrees, below), its default value is $12/\ell$ m^{-1}.

`LookAhead` l_z

The depth control look ahead parameter l_z is input in m. By default, if there is only one `Hull` component, l_z is set to $\ell/2$; otherwise a warning–pause message is issued, with the option to zero l_z.

`PhaseLead` $\alpha_{\,(\text{depth})}$, $\tau_{(\text{depth})}$

The depth phase lead controller parameters default to:

$$\begin{aligned} \alpha_{\,(\text{depth})} &= 3.0 \\ \tau_{(\text{depth})} &= 2.0 \end{aligned}$$

`PID` $k_{1\,(\text{depth})}$, $k_{2\,(\text{depth})}$, $K_{P\,(\text{depth})}$, $K_{I\,(\text{depth})}$, $K_{D\,(\text{depth})}$

The depth PID controller parameters default to:

$$\begin{aligned} k_{1\,(\text{depth})} &= 0.5 \\ k_{2\,(\text{depth})} &= 1.0 \\ K_{P\,(\text{depth})} &= 1.2 \\ K_{I\,(\text{depth})} &= 0.0 \\ K_{D\,(\text{depth})} &= -0.3 \end{aligned}$$

`PitchLimit` θ_L

Depth control pitch limit θ_L is in degrees, and should be positive. Its default value is 10 degrees.

10.2 AutoHead Level 2 Records: HedConGain, HedErrGain, PhaseLead, and PID

`HedConGain` $C_{C\,(\text{heading})}$

Heading control gain $C_{C\,(\text{heading})}$ is in degrees, and should be positive. Since the commanded deflection depends on appendage dihedral and control weight W_{IH} (sections 6.4 and 14.3), this parameter can exceed the maximum deflection on an appendage. Its default value is 20 degrees.

`HedErrGain` $C_{E\,(\text{heading})}$

Heading error gain $C_{C\,(\text{heading})}$ is in degrees^{-1}, and should be positive. Its default value is 0.05 degrees^{-1}.

`PhaseLead` $\alpha_{\,(\text{heading})}$, $\tau_{(\text{heading})}$

The heading phase lead controller parameters default to the same values as for depth control, section 10.1.

`PID` $k_{1\,(\text{heading})}$, $k_{2\,(\text{heading})}$, $K_{P\,(\text{heading})}$, $K_{I\,(\text{heading})}$, $K_{D\,(\text{heading})}$

The heading PID controller parameters default to the same values as for depth control, section 10.1.

10.3 AutoProp Level 2 Records: PhaseLead, PID, PwrErrGain, RevConGain, ThrErrGain, UktErrGain, and UmsErrGain

`PhaseLead` $\alpha_{\,(\text{prop})}$, $\tau_{(\text{prop})}$

The propulsion phase lead controller parameters default to the same values as for depth control, section 10.1.

`PID` $k_{1\,(\text{prop})}$, $k_{2\,(\text{prop})}$, $K_{P\,(\text{prop})}$, $K_{I\,(\text{prop})}$, $K_{D\,(\text{prop})}$

The propulsion PID controller parameters default to:

$$\begin{aligned} k_{1\,(\text{prop})} &= 0.5 \\ k_{2\,(\text{prop})} &= 1.0 \\ K_{P\,(\text{prop})} &= 1.0 \\ K_{I\,(\text{prop})} &= 0.0 \\ K_{D\,(\text{prop})} &= 0.0 \end{aligned}$$

`PwrErrGain` $C_{E\,(\text{prop–power})}$

Power error gain $C_{C\,(\text{prop–power})}$ is in W$^{-1/3}$, and should be positive. Based on a rough estimate of power required for a small submarine at about 1 kt, its default value is 0.12 W$^{-1/3}$.

`RevConGain` $C_{C\,(\text{prop})}$

Propulsion revolutions control gain $C_{C\,(\text{prop})}$ is in RPM, and should be positive. Based on a speed of about 1 kt, its default value is 12 RPM.

`ThrErrGain` $C_{E\,(\text{prop–thrust})}$

Thrust error gain $C_{C\,(\text{prop–thrust})}$ is in $\text{N}^{-1/2}$, and should be positive. Based on a rough estimate of thrust required for a small submarine at about 1 kt, its default value is 0.0375 $\text{N}^{-1/2}$.

`UktErrGain` $C_{E\,(\text{prop–speed})}$

The `UktErrGain` record is used to input the speed error gain $C_{C\,(\text{prop–speed})}$ in kt^{-1}; the value should be positive. Its default value is 1 kt^{-1}.

`UmsErrGain` $C_{E\,(\text{prop–speed})}$

The `UmsErrGain` record is used to input the speed error gain $C_{C\,(\text{prop–speed})}$ in $(\text{m/s})^{-1}$; the value should be positive. Its default value is $1/0.514\ddot{4}$ $(\text{m/s})^{-1}$ (i.e., 1 kt^{-1}).

10.4 AutoRoll Level 2 Records: RolConGain, RolErrGain, PhaseLead, and PID

`RolConGain` $C_{C\,(\text{roll})}$

Roll control gain $C_{C\,(\text{roll})}$ is in degrees, and should be positive. Since the commanded deflection depends on appendage dihedral and control weight W_{IR} (sections 6.4 and 14.3), this parameter can exceed the maximum deflection on an appendage. Its default value is 20 degrees.

`RolErrGain` $C_{E\,(\text{roll})}$

Roll error gain $C_{C\,(\text{roll})}$ is in degrees^{-1}, and should be positive. Its default value is 0.05 degrees^{-1}.

`PhaseLead` $\alpha_{\,(\text{roll})}$, $\tau_{\,(\text{roll})}$

The roll phase lead controller parameters default to the same values as for depth control, section 10.1.

`PID` $k_{1\,(\text{roll})}$, $k_{2\,(\text{roll})}$, $K_{P\,(\text{roll})}$, $K_{I\,(\text{roll})}$, $K_{D\,(\text{roll})}$

The roll PID controller parameters default to the same values as for depth control, section 10.1.

11 Root.geo Level 2 HPAir and MBT Input

The blowing model in `DSSP21` allows representation of a number of typical water ballast system configurations. There can be up to nine HP air bottle groups (BGs) and eight main ballast tanks (MBTs) in the system. Figure 8 illustrates an arrangement of three BGs and three MBTs in which one of the BGs is dedicated to a single MBT, one can

be used with two of them, and the third can be used with any of the three. Isolation of the HP air sources from each other by check valves, as shown in the figure, is assumed. A similar arrangement to figure 8 is found in some submarine systems that employ a main HP blowing source, which in the figure would be BG2, with additional augmentation or emergency sources, BG1 and BG3. In some cases, tanks can only be blown together, for example if MBT1 and MBT2 were connected to BG1 with a single gate valve instead of separate ones. In `DSSP21`, this distinction is not made in the definition of the components, but can be made with `Blow` commands, section 16. Small UUVs typically have simpler systems comprising either a single source or an individual source integrated with each tank.

The connections are modeled in the program by associations between BGs and MBTs. Associations are set up by assigning a type to the components or are otherwise established by default. (Although only one BG of each type is usually necessary to characterize a ballast system, `DSSP21` allows more that one BG of each type.) The rules for association are:

- a BG of type `Main` is associated with all MBTs;
- a BG of type `Forward` or `Aft` is associated with all tanks of the same type; and
- remaining BGs are paired off with remaining MBTs in the order in which they are specified on the *Root*`.geo` file.

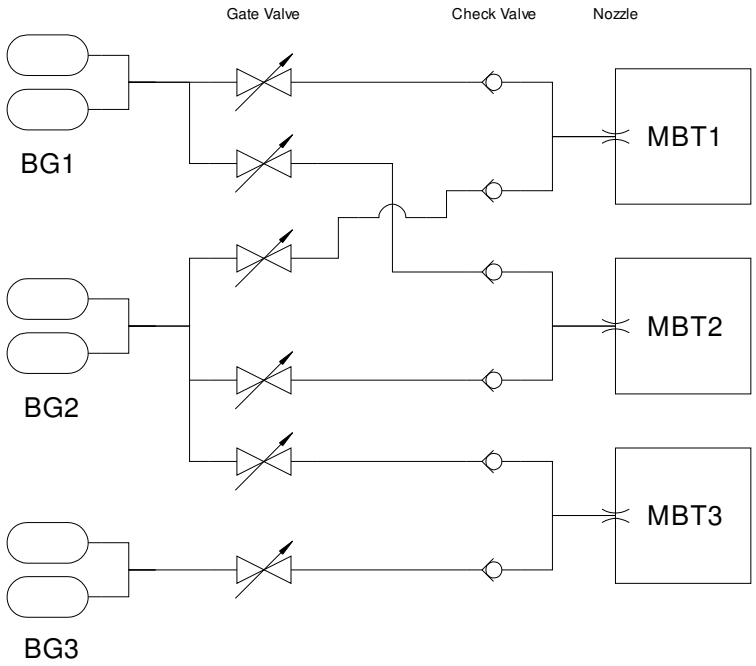


Figure 8. Schematic of connections between three bottle groups and three ballast tanks. BG1 can blow MBT1 and MBT2 individually or together, BG2 can blow any of the three tanks individually or together, and BG3 can blow only MBT3.

For example, the connections of figure 8 can be established by giving BG1, MBT1, and MBT2 type `Forward`, giving BG2 type `Main`, and giving BG3 and MBT3 type `Aft`.

Blowing a tank with more than one BG is computationally inefficient in the current blowing model. If the simulation consists of a single blow of this type, combining the BGs to be used into a single source of the same total volume will avoid the problem. The pressure of the single source, P_{total}, can be obtained from $P_{total}V_{total} = \sum_i P_i V_i$, where i runs over the BGs used. Isothermal and adiabatic blowing are discussed in section 16; the combined BG model gives identical results to blowing with individual BGs isothermally, and quite similar results to blowing with them adiabatically unless the initial pressures are very different.

A bottle group is announced by a `HPAir` record. The following table summarizes the level 2 input that characterizes it:

Keyword	Function
`Aft`	set BG type
`Forward`	set BG type
`Fwd`	set BG type
`Label`	BG label
`Main`	set BG type
`Pop`	force termination of input — see section 5.4
`Pressure`	set BG initial pressure
`Volume`	set BG volume

The `Label` definition is analogous to that of other components. If there is no `Label` record, the default label is `#HPBG`*nnn* where *nnn* is the number of this BG as encountered in the input file. The other records are described in section 11.1.

A ballast tank is announced by a `MBT` record. The following table summarizes the level 2 input that characterizes it:

Keyword	Function
`Aft`	set MBT type
`Forward`	set MBT type
`Fwd`	set MBT type
`Label`	MBT label
`Location`	set location of MBT centroid
`MassFlow`	set initial blowing mass flow into MBT
`Pop`	force termination of input — see section 5.4
`Vent`	set air mass flow venting rate
`Volume`	set MBT blowable volume

The `Label` definition is analogous to that of other components. If there is no `Label`

record, the default label is `#BALT`*nnn* where *nnn* is the number of this MBT as encountered in the input file. The other records are described in section 11.2.

11.1 HPAir Level 2 Records: Aft, Forward, Fwd, Main, Pressure, and Volume

`Aft`

This record sets the BG to type `Aft`. It will be associated with MBTs of type `Aft`.

`Forward`

This record sets the BG to type `Forward`. It will be associated with MBTs of type `Forward`.

`Fwd`

`Fwd` is a synonym for `Forward`. This record sets the BG to type `Forward`, and the BG will be associated with MBTs of type `Forward`.

`Main`

This record sets the BG to type `Main`. It will be associated with all MBTs.

`Pressure` $P_{HP\bullet}$

This record is mandatory. It defines the initial BG pressure, $P_{HP\bullet}$, in bar; pressure must be in the range $100 \leq P_{HP\bullet} \leq 400$ bar.

`Volume` V_{HP}

This record is mandatory. It defines the BG volume, V_{HP}, in m^3.

11.2 MBT Level 2 Records: Aft, Forward, Fwd, Location, MassFlow, Vent, and Volume

`Aft`

This record sets the MBT to type `Aft`. It will be associated with BGs of type `Aft`.

`Forward`

This record sets the MBT to type `Forward`. It will be associated with BGs of type `Forward`.

`Fwd`

`Fwd` is a synonym for `Forward`. This record sets the MBT to type `Forward`, and the MBT will be associated with BGs of type `Forward`.

`Location` x_{BT} , y_{BT} , z_{BT}

This record is mandatory. It defines the coordinates, in vehicle axes, of the centroid of the MBT. In the present build, changes of mass resulting from a blow are assumed to occur at this point.

`MassFlow` $\dot{m}_{BT\bullet}$

This record is mandatory. It defines the initial blowing mass flow rate of air into the MBT, in kg/s. At typical HP air pressures, this quantity is determined by the nozzle that releases air into the tank.

`Vent` $\dot{m}_{BV}$

This record defines the venting mass flow rate of air from the MBT, in kg/s. The rate is constant for the duration of the simulation, independent of depth. In the absence of this record, $\dot{m}_{BV} = 0$ (no venting).

`Volume` V_{BT}

This record is mandatory. It defines the blowable volume of the MBT, V_{BT}, in m^3.

12 Root.geo Level 2 WTC Input

There can be up to eight watertight compartments (WTCs) defined in the vehicle; they are required for modeling a flood during a `Free` simulation. Each watertight compartment is announced by a `WTC` record, and defined by subsequent level 2 input.

A WTC is assumed to be a circular cylinder, defined by diameter, length, the location of its center, and occupancy: the fraction of its volume that is already occupied, and therefore not available for flooding. The moment due to floodwater is estimated as a function of both flooded volume and pitch angle.

The following table summarizes `WTC` level 2 input:

Keyword	Function
`Diameter`	set WTC diameter
`Label`	WTC label
`Length`	set MBT length
`Location`	set location of WTC center point
`Occupancy`	set WTC occupancy
`Pop`	force termination of input — see section 5.4

The `Label` definition is analogous to that of other components. If there is no `Label` record, the default label is `#WTCP`*nnn* where *nnn* is the number of this WTC as encountered in the input file. The other records are described in section 12.1.

12.1 WTC Level 2 Records: Diameter, Length, Location, and Occupancy

`Diameter` D_C

This record is mandatory. The argument D_C must be positive. Note that this dimension is not checked for consistency with any other input (such as a hull diameter).

`Length` L_C

This record is mandatory. The argument L_C must be positive. Note that this dimension is not checked for consistency with any other input (such as a hull length).

`Location` x_C , y_C , z_C

This record is mandatory. It defines the coordinates, in vehicle axes, of the centroid of the unoccupied cylinder modeling the WTC. Note that this point is not checked for consistency with any other input (such as hull location and dimensions).

`Occupancy` O_C

This defines the occupancy O_C of the WTC. The maximum amount of floodwater that can enter is $\frac{\pi}{4} {D_C}^2 L_C (1 - O_C)$. The argument is mandatory; it must be in the range $0.05 \leq O_C \leq 0.95$. In the absence of this record, occupancy defaults to $O_C = 0.3$.

13 Root.geo Level 2 Plot Input

If a `Plot` record is found at level 1, plotting output of the vehicle geometry is saved in ASCII format on file *Root*`.gpl`. A plot enables the user to check that the geometry has been input correctly. Component intersections should be checked with a wire frame plot using no hidden lines so that the details are not obscured.

Plotting requires representation of the component cross sections, which are artefacts not defined or used elsewhere in the program. Hull cross sections are represented by an ellipse fitted to B_i and T_i; the number of perimeter segments is defined by a parameter n_{segH}. Lifting component cross sections, at the root and tip only, are represented by a NACA 4-digit section fitted to their chord and maximum thickness; the number of perimeter segments is defined by a parameter n_{segL}.

The format of *Root*`.gpl` is controlled by a number of optional input records at level 2:

Keyword	Function
`AcroSpin`	format *Root*`.gpl` for Acrospin software
`HullSegmnt`	set hull section plotting parameter

`LiftSegmnt`	set lifting appendage section plotting parameter
`Maple`	format *Root*`.gpl` for the Maple procedure `DSSPplot.mpl`
`MatLab`	format *Root*`.gpl` for the MATLAB script `DSSPplot.m`
`VRML`	output *Root*`.gpl` in VRML 1 format

The *Root*`.gpl` default file format is VRML 1.

13.1 AcroSpin, HullSegmnt, LiftSegmnt, Maple, MatLab, and VRML Records

`AcroSpin`

Root`.gpl` is formatted for AcroSpin Version 2.0 for DOS. This is an old program; the file format is compatible with the 'original DOS version of Acrospin', actually version 2.3, that was at one time marketed as a viewer for MathWare Derive version 5 (`http://www.mathware.com`), but appears to be no longer available. AcroSpin produces a wire-frame screen plot with fast zoom, rotation, and translation capabilities. An example screen shot is shown in figure 9.

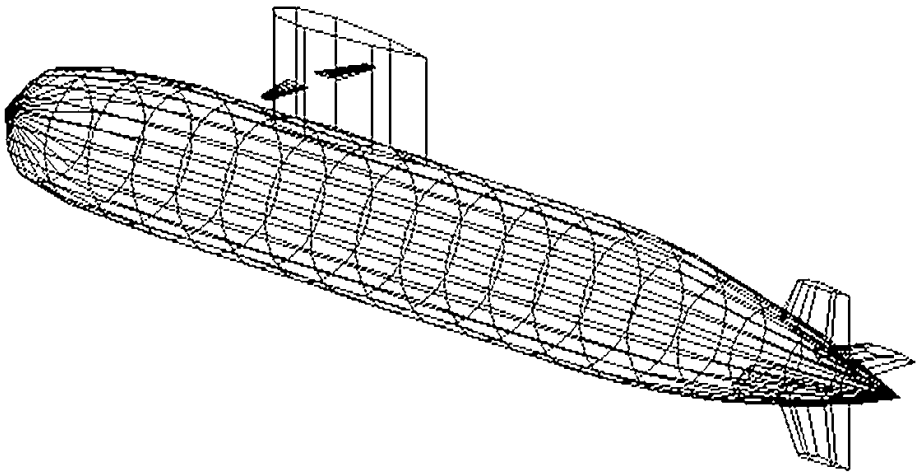


Figure 9. `AcroSpin` plot screen shot produced from the example input in annex D.

`HullSegmnt` n_{segH}

This record overrides the default value of n_{segH}, the number of hull cross section segments. The argument is mandatory and must be one of 4, 6, 8, 12, or 24. By default, $n_{segH} = 24$.

`LiftSegmnt` n_{segL}

This record overrides the default value of n_{segL}, the number of lifting component cross section segments. The argument is mandatory and must be one of 2, 4, or 8. By default, $n_{segL} = 8$.

`Maple`

Root`.gpl` is formatted for a Maple (`http://www.maplesoft.com`) procedure, `DSSPplot.mpl`, written by George Watt at DRDC. Options include solid modeling (patch plot) or wire frame with zoom, rotation, translation, and perspective capabilities. A basic patch plot is shown in figure 10.

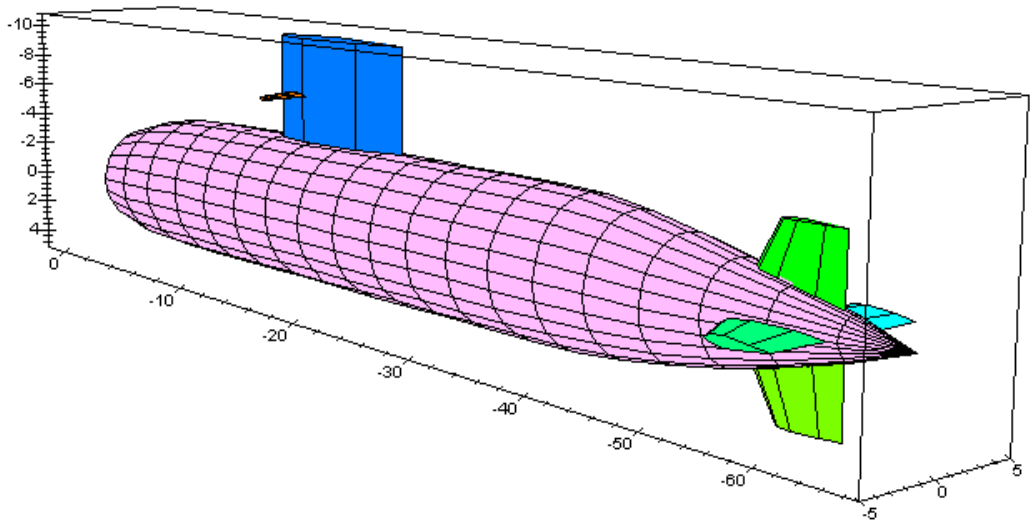


Figure 10. `Maple` plot screen shot produced from the example input in annex D; component colour is related to its input order, not type. The axis box is optional.

`MatLab` [*keyword*]

Root`.gpl` is formatted for a MATLAB (`http://www.mathworks.com`) script, `DSSPplot.m` (written by the author for MATLAB Version 6.0, Release 12). The optional keyword argument may be `Colour`, in which case the result is a patch plot with patches that are yellow on hull components, cyan on non-controlling appendages, and red on controlling appendages. The default is `White`, in which the patches are white and the standard lighted view appears in greyscale. The white patch image also permits a wireframe view by the user invoking 'hidden off' at the MATLAB command line: hidden line removal is turned off and the patches are made transparent. This does not work with coloured patches.

To examine the image it is easiest to invoke 'Camera Toolbar' from the figure window View menu, and then use the camera controls to orient the image and to adjust perspective, viewpoint, etc.

Example screen-shots are shown in figures 11 and 12.

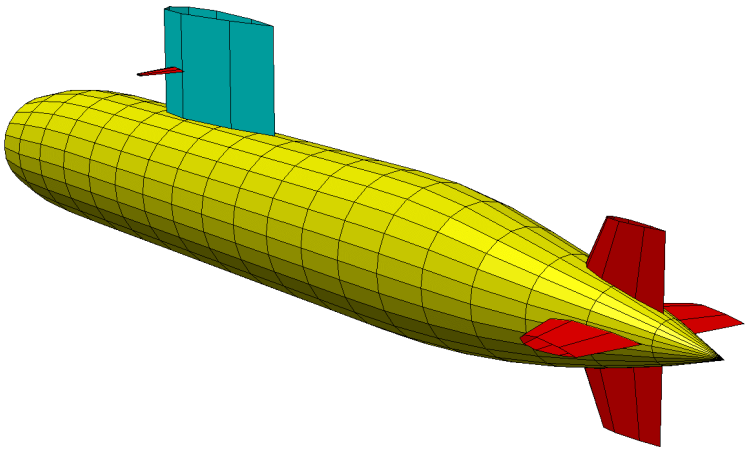


Figure 11. `MatLab` plot screen shot produced from the example input in annex D using the keyword `Colour`; the axis box is hidden.

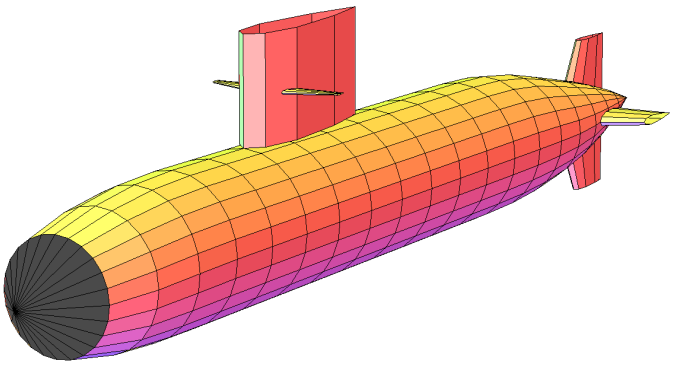


Figure 12. `MatLab` plot screen shot produced from the example input in annex D using the (default) keyword `White` and additional external lighting: red to port, green to starboard, yellow above, and blue below.

`VRML` $[n_{version}]$

Root`.gpl` is output in VRML 1 format if $n_{version} = 1$. In future builds of the code, VRML 2 format will be available if $n_{version} = 2$. The argument is optional; VRML 1 is used by default. Components are coloured: hulls are yellow, non-controlling appendages are cyan, and controlling appendages are red.

VRML was developed in the mid-1990s as a universal modeling standard. While it did not reach that status, it is still supported by a number of rendering programs and web browser plug-ins, many of which are free-ware — although some require licensing for certain types of commercial use. The example screen-shots shown in figures 13 and 14 were produced with the free ModelPress Reader (Informative Graphics Corporation, `http://www.modelpress.com`). A useful feature of this program for checking geometry is the ability to easily display the distance between nodes (panel corner points) on the model, figure 13.

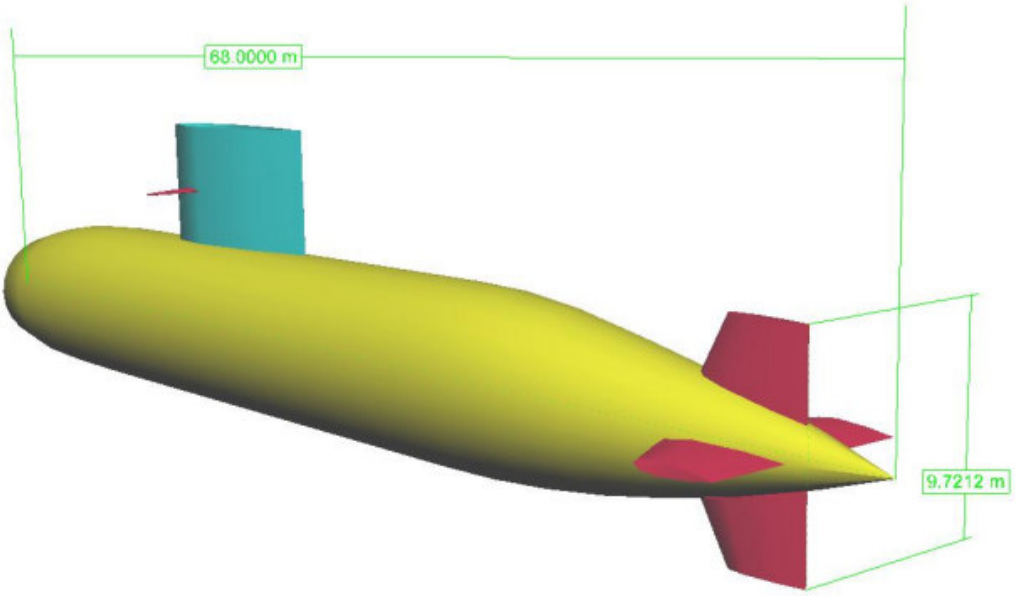


Figure 13. ModelPress Reader screen shot produced with VRML 1 output from the example input in annex D, illustrating shaded rendering with measurements.

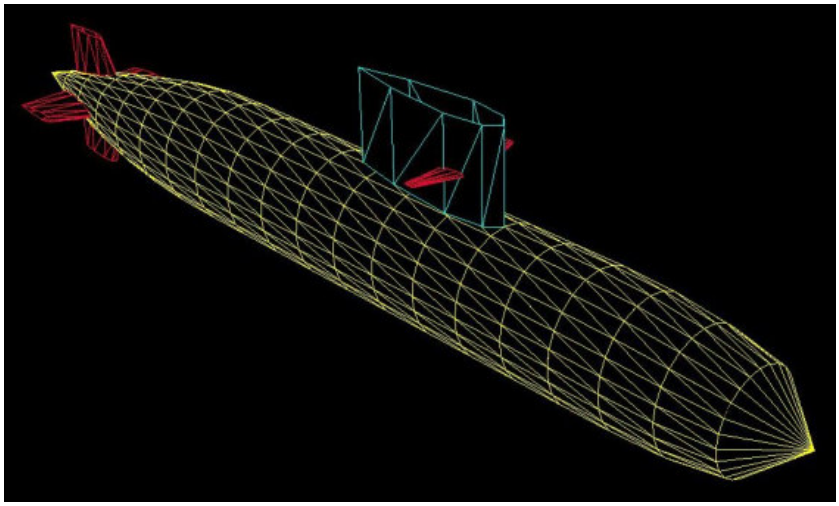


Figure 14. ModelPress Reader screen shot produced with VRML 1 output from the example input in annex D, illustrating hidden-line wire frame rendering on the default black background. Panels are subdivided into triangles for the shading option in the Reader.

14 Root.sim Control Deflection Commands

Control deflections are commanded in *Root*`.sim` by the keyword `Delta` or its synonym `Del`. They may also be controlled by one or more autopilots as described in section 10. Internally, the program handles each control appendage as an independent entity; this is different from the way that the operator commands deflection of the rudders, sternplanes, etc., and may be different again from how a heading or depth control autopilot is implemented. In order to accommodate these differences, several control deflection command modes are available.

The three principal modes of command can be summarized as:

- direct input, which allows the user to specify an individual appendage to be deflected,
- quasi-direct input, which indicates the type(s) of appendages and implied sense of their deflection, and
- indirect input, which indicates the desired response of the vehicle and equivalent magnitude of the deflection.

There are also a few global control input commands.

Control deflections in a static calculation and commanded deflections in a manoeuvering simulation remain set until they are changed by another `Delta` command. Under

autopilot control in a manoeuvering simulation, on the other hand, the commanded deflections, which are in indirect mode, are dynamic. To balance these different requirements, the modes of deflection command are limited in how they work together. In addition, indirect input sets flags that allow only continued indirect input to the affected appendages until canceled by a `Clear` command. This has its origin in requirements for autopilot control: a direct or quasi-direct command is not permitted to 'kick-out' an autopilot. In general, multiple deflection commands to the same appendage are not permitted; however, some combinations of multiple indirect commands are allowed. Command conflicts are discussed in section 14.5.

14.1 Direct Deflection Input

The command is directed at a specific appendage:

`Delta` *label string* or N_L , δ or *keyword*

If the second argument is δ, a control deflection of δ degrees in local coordinates is applied to the lifting appendage identified by the label string or by the integer N_L. Attention must be given to sign because of the local coordinate convention (section 3.4). For example, to deflect horizontal sternplanes to generate upward force, the starboard plane requires positive δ while the port plane requires negative δ.

Alternatively, the second argument to `Delta` can be a keyword, specifically one of `Clear`, `Reset`, or `Zero`:

`Clear` : the control surface is returned to its initial deflection and control flags are cleared. The primary use of this command is to cancel indirect input. Unlike `Reset`, below, `Clear` can be combined with another control deflection command: regardless of order, `Clear` is executed first, then the other command.

`Reset` : the control surface is returned to its initial deflection and control flags are cleared. `Reset` cannot be combined with another control deflection command.

`Zero` : the control surface is set at zero deflection and control flags are cleared. `Zero` cannot be combined with another control deflection command.

14.2 Quasi-Direct Deflection Input

The command is applied to one or more appendage types that are individually associated with a direction of the resulting control force. The notation $\pm\delta$ used below indicates that the sign of an appendage deflection depends on that direction; its magnitude is $|\delta|$.

`Delta` *keyword* , δ

The keyword is one of:

`Bowplane` : appendages with type `LBowplane` and `SBowplane` are deflected by $\pm\delta$ to produce an upward force.

`Foreplane` : appendages with type `LBowplane`, `SBowplane`, and `Sailplane` are deflected by $\pm\delta$ to produce an upward force.

`Rudder` : appendages with type `Rudder` are deflected by $\pm\delta$ to produce a force to starboard.

`Sailplane` : appendages with type `Sailplane` are deflected by $\pm\delta$ to produce an upward force.

`Sternplane` : appendages with type `Sternplane` are deflected by $\pm\delta$ to produce an upward force.

In this list, the resultant force direction is indicated for positive δ; it will be opposite for negative δ.

Quasi-direct input is appropriate for a cruciform arrangement of the aft appendages, and for horizontal foreplanes. X-rudders and other non-cruciform arrangements should have the type `Tail`, which does not respond to quasi-direct deflections.

14.3 Indirect Deflection Input

Indirect commands affect all control appendages that can contribute to the vehicle response indicated by the keyword. The magnitude of the deflection is weighted individually for each appendage according to W_{ID}, W_{IH}, W_{IR} introduced in section 6.4, and takes account of its anhedral k_ϕ, defined below. The notation $\pm\ldots\delta$ is similar to that in the previous section; it indicates a deflection of $|\ldots\delta|$ with the appropriate sign for the appendage.

`Delta` *keyword* , δ

Where the keyword is one of:

`Down` : appendages with type `LBowplane`, `SBowplane`, and `Sailplane` are deflected by $\pm W_{ID} k_\phi \delta$ to produce a downward force, and appendages with type `Sternplane` and `Tail` are deflected by $\pm W_{ID} k_\phi \delta$ to produce an upward force.

`Port` : appendages with type `Rudder` and `Tail` are deflected by $\pm W_{IH} k_\phi \delta$ to produce a force to starboard.

`RollNeg` : appendages with type `Tail` are deflected by $\pm W_{IR} \delta$ to produce a negative roll moment (i.e., counterclockwise looking forward).

`RollPos` : appendages with type `Tail` are deflected by $\pm W_{IR} \delta$ to produce a positive roll moment (i.e., clockwise looking forward).

`Stbd` : as `Starboard`, below.

`Starboard` : appendages with type `Rudder` and `Tail` are deflected by $\pm W_{IH} k_\phi \delta$ to produce a force to port.

`Up` : appendages with type `LBowplane`, `SBowplane`, and `Sailplane` are deflected by $\pm W_{ID} k_\phi \delta$ to produce an upward force, and appendages with type `Sternplane` and `Tail` are deflected by $\pm W_{ID} k_\phi \delta$ to produce a downward force.

`Stbd` is a synonym for `Starboard`; making it a keyword allows the common abbreviation `Stbd` to be used without ambiguity. The vehicle response indicated is for a positive value of δ; the response is opposite if δ is negative. Therefore, '`Up +10`' produces the same result as '`Down -10`'. For `Up`, `Down`, `Port`, and `Stbd` the commanded deflection is multiplied by anhedral factor $k_\phi = \cos\phi$, where ϕ is appendage anhedral in the appropriate crossflow direction (horizontal or vertical).

Indirect deflection commands are processed independently according to whether they are associated with depth control (`Up` and `Down`), heading control (`Port` and `Stbd`), or roll control (`RollNeg` and `RollPos`). Although multiple input in any one mode is not permitted, mixed indirect input is allowed. Thus, for X-rudders with type `Tail`, the following mixed input:

```
Del Stbd 10
Del Down  5
```

will result in the rudders being deflected so as to produce both responses simultaneously.

An appendage under indirect control requires a `Clear` command before it can be given a direct or quasi-direct deflection command.

14.4 Global Deflection Input

`Delta` `All` , *keyword*

The only valid first argument for global input is the keyword `All`; more options may be added in future versions of the code.

The second keyword must be one of:

`Clear` : all control surfaces are returned to their initial deflection and control flags are cleared. The primary use of this command is to cancel indirect input. Unlike `Reset`, below, `Clear` can be combined with other control deflection commands; regardless of order, the global `Clear` is executed first, then the other commands.

`Reset` : all control surfaces are returned to their initial deflection and control flags are cleared. `Reset` cannot be combined with any other control deflection commands.

`Zero` : all control surfaces are set at zero deflection and control flags are cleared. `Zero` cannot be combined with any other control deflection commands.

14.5 Conflicting Deflection Commands

`DSSP21` does not permit deflection command conflicts. Between static runs, or simultaneously for dynamic simulations, only the following combinations of deflection commands are allowed for any individual appendage:

- a direct command with `Clear`,
- a quasi-direct command with `Clear`,
- an indirect-depth, indirect-heading, and indirect-roll command with each other, and with `Clear`,
- `Reset`, or `Zero`.

A fatal error is generated for any other combination.

Commanded deflections stay in effect until legitimately canceled or replaced by a new command. An appendage under indirect control requires a `Clear` command before or coincident with returning it to direct or quasi-direct control. Failure to do so generates a fatal error. In a free manoeuvering simulation, active depth, heading, and roll autopilots use indirect deflection control, and the same rule applies. Issuing a `Stop` command to an active autopilot cancels the indirect commands associated with it.

15 Root.sim Heading Commands

The commands discussed in this section are used in a free manoeuver to define initial compass heading and to interpret commands to the heading autopilot. While some of the options in this section are presently redundant, they may be required in a compass heading description of the trajectory for future developments. It may be useful to refer to figure 2. Heading angle increases in a turn to starboard.

Within `DSSP21`, instantaneous heading is the azimuthal angle ψ relative to the initial heading (i.e., $\psi_\bullet = 0$). In a free manoeuver, this is tabulated on `sim`*nnn*`.txt` together with the equivalent compass heading based on an initial compass heading input by the user, section 22.2. By default this initial heading is zero degrees, or North. Commanded heading can be defined in either system. Heading input comprises a string of up to three words that are treated as a single argument in the commands that start the heading autopilot or set the initial compass heading.

For a heading autopilot command, neither the current heading ψ nor the initial compass heading is necessarily zero. The commanded heading is then evaluated from the argument string as follows, where *value* is numerical:

a. *value* : azimuthal degrees relative to the initial reference.
b. *value* `Points` : azimuthal angle in points relative to the current heading.
c. *value* `Port`/`Stbd`/`Starboard` : degrees to port or starboard of the current heading.
d. *value* `Points` `Port`/`Stbd`/`Starboard` : points to port or starboard of the current heading.
e. *value* `Relative` : degrees relative to the current heading.
f. *value* `Azimuth` : degrees azimuth on the compass.
g. `N`/`North`/`NbyE`/.../`NbyW` : compass heading. See annex B for the valid keywords; they may not be abbreviated for input.
h. `North` *value* `East`, `N` *value* `W`, etc. : compass quadrant heading in degrees. The keywords may not be abbreviated for input.

Note that `DSSP21` outputs current heading in the range $-180° \leq \psi \leq 180°$ if quaternions are used to represent the Euler angles (section 22.3), but heading is unbounded otherwise. Compass heading is always output in the range 0 to 360 degrees.

The following examples illustrate commanded (subscript c) heading resulting from the various input formats. It is assumed that the initial compass heading in the manoeuver was `East`, and that the current heading ψ is 20 degrees (i.e., a compass heading of 110 degrees).

Format	Argument String	ψ_c	Compass_c
a.	−25	−25.00°	65.00°
b.	2 `Points`	42.50°	132.50°
c.	15 `Port`	5.00°	95.00°
d.	3 `Points Port`	−13.75°	76.25°
e.	−25 `Relative`	−5.00°	85.00°
f.	−25 `Azimuth`	−115.00°	335.00°
g.	`NWbyW`	−146.25°	303.75°
h.	`S 40 E`	50.00°	140.00°

While a great deal of flexibility is provided in these options, it should be obvious that mixing commands based on current heading and compass heading in the same input file is a potential source of confusion; users should exercise due care.

The compass heading and the current heading default to zero. Therefore, for the purpose of inputting a specific initial heading, formats a., e., and f. are exactly equivalent to each other, and the values in each of formats a. to f. represent the compass heading relative to zero (North).

16 Root.sim MBT Blowing and Venting Commands

Ballast blowing is commanded during a free manoeuver by the keyword `Blow` (section 22.4), which is followed by one or two arguments. The first identifies the tank or tanks to be blown, the second identifies the HP air bottle group(s) to be used. The second argument is optional since blowing can be determined by BG-MBT associations, section 11, if it is not specified otherwise.

The blowing command is:

`Blow` *argument 1* , [*argument 2*]

where *argument 1* is one of:

N_B: the index of the MBT to be blown (the $N_B{}^{th}$ tank defined on *Root*`.geo` has the index N_B).

label string: the label of the MBT to be blown.

`Aft`/`Forward`/`Fwd`: all MBTs of the type indicated are to be blown.

`All`: all MBTs are to be blown.

`Stop`: stop blowing all MBTs.

and the optional *argument 2* is one of:

N_G: the index of the BG to be used (the $N_G{}^{th}$ bottle group defined on *Root*`.geo` has the index N_G).

label string: the label of the BG to be used.

`Aft`/`Forward`/`Fwd`/`Main`: all BGs of the type indicated are to be used.

`All`: all BGs associated with the tank(s) identified by *argument 1* are to be used.

`Stop`: stop blowing the tank(s) identified by *argument 1*.

The default for *argument 2* is `All`. The following examples illustrate how these arguments are combined:

`Blow 1`	blow MBT 1 with all associated BGs
`Blow 2 Fwd`	blow MBT 2 with BGs of type `Forward`, if associated
`Blow All Main`	blow all MBTs with `Main` BGs
`Blow Aft`	blow all MBTs of type `Aft` with `Aft` BGs
`Blow 3 Stop`	stop blowing MBT 3
`Blow Stop`	is the same as `Blow All Stop`

Care has to be exercised with blowing commands because BGs and MBTs are related only by simple association. If there is any doubt what the associations are, refer to the table that is output on *Root*`.spr` for a free simulation. In addition there are ambiguities: in section 11 it was noted that a bottle group may be connected so that it can only blow two tanks together, as is BG1 in figure 15, although a simple association does not

reflect that. Therefore, we would represent the system in figure 15 the same way as for figure 8: by giving BG1, MBT1, and MBT2 type `Forward`, giving BG2 type `Main`, and giving BG3 and MBT3 type `Aft`; blowing MBT1 and MBT2 simultaneously from just BG1 is achieved with '`Blow Fwd Fwd`' (or '`Blow All Fwd`'). However, there is nothing to prevent the user from using '`Blow 1`' or other commands that may inadvertently result in BG1 blowing MBT1 alone, which is consistent with figure 8 but not with figure 15.

The present build of the program allows two types of ballast blows: isothermal, which is a common approximation in simulation codes, and adiabatic, the default; they will discussed in more detail in the future companion document to this guide. The author is not aware of any reliable data for validation of either method, but it is believed that an isothermal blow overestimates the blowing rate, whereas the adiabatic blow as implemented in `DSSP21` likely underestimates it. Adiabatic blowing is the default because it appears closest to reality while being conservative with respect to the effectiveness of the blow. Isothermal blowing is initiated by the keyword `Isothermal`.

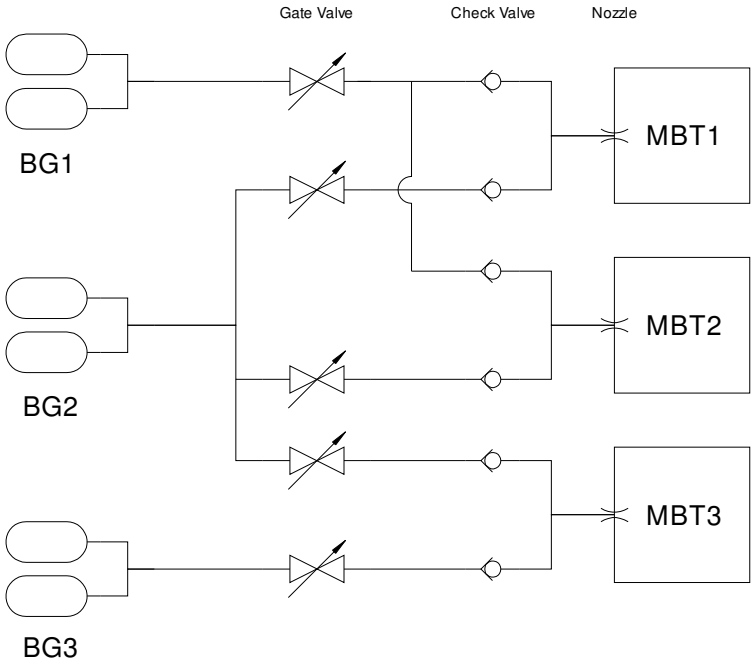


Figure 15. A variation on figure 8: BG1 can only blow MBT1 and MBT2 together, BG2 can blow any of the three individually or together, and BG3 can blow only MBT3.

The command to vent a ballast tank is simpler: it is assumed that air is vented at the rate is set in MBT level 2 input, section 11.2; this rate is constant and independent of depth. There is no heat transfer associated with venting.

The venting command is:

`Vent` *argument 1* , [*argument 2*]

where *argument 1* is one of:

N_B: the index of the MBT to be vented (the $N_B{}^{th}$ tank defined on *Root*`.geo` has the index N_B).

label string: the label of the MBT to be vented.

`All`: all MBTs are to be vented.

`Stop`: stop venting all MBTs.

and optional *argument 2* is :

`Stop`: stop venting the tank(s) identified by *argument 1*.

17 Root.sim WTC Flooding Commands

A flood in a watertight compartment (WTC) is initiated during a free manoeuver by the keyword `Flood` (section 22.4), which is followed by one, two, or three arguments. The first argument identifies the WTC to be flooded, and the second and third are parameters for the breach that the flood originates from.

The flooding record is:

`Flood` *argument 1* , [*argument 2* , [*argument 3*]]

where *argument 1* is one of:

N_C: the index of the WTC to be flooded (the $N_C{}^{th}$ WTC defined on *Root*`.geo` has the index N_C).

label string: the label of the WTC to be flooded.

`Stop`: secure floods in all WTCs.

optional *argument 2* is one of:

D_F: the diameter of the breach, which is assumed to be circular (otherwise, use the diameter of a circle with the same area as the breach).

`Stop`: secure the flood in the WTC identified by *argument 1*.

and optional *argument 3* is:

C_F: the coefficient of discharge of the breach. By default $C_F = 0.6$.

If the first argument is a WTC index or label then the second argument is mandatory. The third argument is only used if the second is D_F. Unless the user has information about the value to use for a specific system breach, *argument 3* can be omitted; the default coefficient of discharge will give acceptable results for most purposes.

The following examples illustrate how the arguments are combined:

`Flood Stop`	secure all floods in progress
`Flood 2 0.08 0.45`	flood in WTC 2 with $D_F = 0.08$ and $C_F = 0.45$
`Flood #ControlRoom Stop`	secure flood in labeled WTC

Note that '`Flood 1 0`' is the same as '`Flood 1 Stop`'.

If a new flood is started in a watertight compartment where one is already in progress, flooding continues with the new parameters D_F and C_F. Therefore, to represent an additional flood in the same WTC, the value of D_F should correspond to the combined area of the existing and new breaches.

18 Root.sim Level 1 Input

Root`.sim` level 1 input defines a sequence of calculations or simulations to be run. Each run is defined by a block of level 2 input records, and the run is executed before control is passed back to level 1. There may be up to 999 runs in the sequence. An example of *Root*`.sim` is given in annex E.

The *Root*`.sim` keyword records at level 1 are:

Keyword	Function
`Captive`	run captive vehicle manoeuvering load calculations
`Coef`	generate a set of hydrodynamic coefficients
`Coefficnt`	generate a set of hydrodynamic coefficients
`Free`	run a free-swimming vehicle simulation
`Help`	send help output to *Root*`.log`
`MLD`	generate Manoeuvering Limitation Diagram curves (inactive)
`Static`	run a set of static load calculations
`Text`	title or other information

18.1 Calculation and Simulation Records: Captive, Coef, Coefficnt, Free, MLD, and Static

These records announce level 2 input for various calculations and simulations. Execution of each one adds to *Root*`.spr` output and generates a new `sim`*nnn*`.txt` output file. A `Free` simulation also adds a new `aux`*nnn*`.txt` file. `Coef` is a synonym for `Coefficnt`.

`Captive`

A `Captive` simulation estimates hydrodynamic load time histories for predetermined vehicle manoeuvers. Its original purpose was to be employed as a diagnostic tool to interpret `Free` manoeuvers, but it can also be used, for example, to estimate hydrodynamic balance loads in MDTF captive model experiments. The level 2 input records are described in section 21.

`Coef`

From a series of static load calculations, generates hydrodynamic coefficients and optionally makes small perturbation stability and plane reversal estimates. The level 2 coefficient input records are described in section 20.

`Coefficnt`

As `Coef`, above.

`Free`

This keyword initiates the time simulation of a single free-swimming vehicle that is manoeuvering in response to a series of control deflection and propulsor commands. The level 2 input records are described in section 22.

`MLD`

This option is a placemarker for future development; it will be used to estimate MLD curves from a series of free-swimming vehicle simulations.

`Static`

This keyword initiates calculation of the loads on a captive vehicle under static conditions, e.g., a model in a fixed-attitude towing tank or rotating arm experiment. The level 2 input records are described in section 19.

In the following sections, level 2 input is described in order of complexity of the calculations required: `Static`, `Coefficnt`, `Captive`, and `Free`.

If the first run on *Root*`.sim` does not include redefinition of water density, a warning is issued to remind the user that the default value is 1000, for fresh water.

18.2 Help and Text Records

`Help`

When the `Help` record is encountered, help text is sent to *Root*`.log`, and the program is stopped. This provides a quick reference for the user if this guide is not readily available; its principal components are dictionary listings and a user input summary.

`Text` *text string*

A `Text` record provides titles and other information; each one is output to *Root*`.spr` as it is encountered.

19 Root.sim Level 2 Static Input

Static hydrodynamic loads are calculated for a given state vector $(u, v, w, p, q, r, \delta)^T$, where δ is a generic control surface deflection. The translational velocities u, v, w can be defined indirectly using incidence angles α, β and total velocity U. Multiple state inputs can be generated by making one parameter per run the argument of a `Loop` record. A set of steady forces and moments for each state is output on `sim`*nnn*`.txt`.

The following table summarizes the keyword records for level 2 static run input:

Keyword	Function
`Alfa`	define angle of attack
`Alpha`	define angle of attack
`Beta`	define angle of drift
`Del`	set control appendage deflection(s)
`Delta`	set control appendage deflection(s)
`DeltaLim`	reset control appendage deflection limits
`Loop`	iterate a parameter between limits
`P`	set state variable p
`Pop`	force termination of input — see section 5.4
`Q`	set state variable q
`R`	set state variable r
`Reference`	redefine reference origin and axes
`Rho`	redefine fluid density
`Text`	output run information or other descriptive text
`U`	set state variable u
`Ukt`	set total velocity in kt
`Ums`	set total velocity in m/s
`V`	set state variable v
`W`	set state variable w
`WeightDCI`	reset indirect control deflection weights

`Alfa` and `Alpha` are synonyms.

19.1 Direct State Variable Records: Del, Delta, P, Q, R, U, V, and W

`Del` Delta arguments: see section 14

Any of the direct, quasi-direct, indirect, or global `Delta` arguments discussed in section 14 can be used on this record. Multiple `Del` records can be present so long as a command conflict, section 14.5, is avoided.

`Delta` Delta arguments: see section 14

As `Del`, above.

`P` p

The mandatory argument is roll rate in deg/s. The default is zero.

`Q` q

The mandatory argument is pitch rate in deg/s. The default is zero.

`R` r

The mandatory argument is yaw rate in deg/s. The default is zero.

`U` u

The mandatory argument is axial velocity (surge) in m/s. A `U` record must be present for direct translational input; on the other hand, one cannot appear in the same `Static` run as indirect translational input, section 19.2.

`V` v

The mandatory argument is lateral velocity (sway) in m/s. The default is zero. A `V` record cannot appear in the same `Static` run as indirect translational input, section 19.2.

`W` w

The mandatory argument is normal velocity (heave) in m/s. The default is zero. A `W` record cannot appear in the same `Static` run as indirect translational input, section 19.2.

19.2 Indirect State Variable Records: Alfa, Alpha, Beta, Ukt, and Ums

`Alfa` α

The mandatory argument is angle of attack, $\alpha = \tan^{-1}(w/u)$, in degrees. The default is zero. An `Alfa` record cannot appear in the same `Static` run as direct translational input, section 19.1.

`Alpha` α

As `Alfa`, above.

`Beta` β

The mandatory argument is angle of drift, $\beta = \tan^{-1}(-v/u)$, in degrees. The default is zero. A `Beta` record cannot appear in the same `Static` run as direct translational input, section 19.1.

`Ukt` U

The mandatory argument is total velocity, $U = \sqrt{u^2 + v^2 + w^2}$, in kt; the default value is 1.944 kt (1 m/s). A `Ukt` record cannot appear in the same `Static` run as direct translational input, section 19.1, or a `Ums` record.

`Ums` U

The mandatory argument is total velocity, $U = \sqrt{u^2 + v^2 + w^2}$, in m/s; the default value is 1 m/s. A `Ums` record cannot appear in the same `Static` run as direct translational input, section 19.1, or a `Ukt` record.

19.3 The Loop Record

The `Loop` record is used to iterate a state parameter between limits. A typical application would be to cover a range of incidence angles, although any of the parameters listed in sections 19.1 and 19.2 can be iterated. For a parameter Q, the syntax of this record is:

`Loop` *Argument(s)* , Q_{From} , Q_{To} , $[\Delta Q]$

where numerical values Q_{From} and Q_{To} are the limits between which parameter Q is to be iterated, and ΔQ is the increment for iteration. A single first argument, which must be one of `Alfa`, `Alpha`, `Beta`, `P`, `Q`, `R`, `U`, `Ukt`, `Ums`, `V`, or `W`, identifies parameter Q. However, if the first argument is `Del` or `Delta`, an additional argument is required: a label string or integer index for a direct command, or an appropriate keyword for a quasi-direct or indirect command, as described in sections 14.1 to 14.3.

The following are examples of valid `Loop` records:

```
Loop Beta -30 30 2
Loop Delta Up -10 10 .5
Loop del 3 -10 15 2.5
Loop ukt 0.5 5 0.5
loop q -1 2
loop r -6  12 -2
```

The increment has been omitted from the penultimate example; this results in the loop executing just the Q_{From} and Q_{To} values. If Q_{From} and Q_{To} are inconsistent with the sign of ΔQ, they are interchanged, so the last example above will be executed as if it were:

```
loop r 12 -6 -2
```

$Q_{To}-Q_{From}$ need not be an integer multiple of the increment. The number of increments is set to most closely approach Q_{To} on the final iteration.

A loop takes on the characteristics of parameter Q: in particular, whether Q is a direct or indirect state variable. For example, if Q is r (i.e., `R`, direct), speed must be defined

by a `U` record, whereas if Q is α (`Alpha`, indirect), speed must be defined, if required, with a `Ukt` or `Ums` record.

There can be only one `Loop` record in a `Static` run.

19.4 DeltaLim, Reference, Rho, Text, and WeightDCI Records

These records can be input for most of the calculations and simulations on *Root*`.sim`. `DeltaLim`, `Reference`, `Rho`, and `WeightDCI` redefine basic vehicle and simulation parameters that then remain unchanged until redefined in a subsequent run. A `Text` record is simply output to *Root*`.spr` when it is encountered.

`DeltaLim` *label string* or N_L , δ_{Max} , δ_{Min}

Resets control deflection limits on the lifting appendage identified by the label string or by the integer N_L, so that $\delta_{Max} \geq \delta \geq \delta_{Min}$. The appendage must be either `AllMoving` or `Flapped`. Deflection limits on a particular appendage cannot be reset more than once per run, but multiple `DeltaLim` records can be used to reset limits for different appendages.

`Reference` [*keyword*]

Redefines the vehicle reference axes and origin. If the argument is keyword `CB` or `CG` the reference origin is located at the CB or CG of the vehicle with the reference axes directed along the vehicle axes. Otherwise, a reference origin and axes must follow in a data block as two triplets of real numbers; the full syntax is given in section 4.5. Despite the recommendation in section 3.4 against a drastic redefinition of the reference, there are no serious consequences from doing so in a `Static` or `Captive` run. However, there may be numerical difficulty in subsequent `Free` runs since they inherit the previous reference by default. `Coefficnt` calculations will be incorrect if the reference is moved to a location without port-starboard symmetry.

Angular motions and calculated moments are with respect to the redefined reference.

`Rho` ρ

Redefines fluid density; the argument should be in the range $990 \leq \rho \leq 1035$ (kg/m^3).

`WeightDCI` *keyword* , *keyword* or

`WeightDCI` *label string* or N_L , W_{ID} , W_{IH} , [W_{IR}]

Resets indirect control weights on all lifting appendages if the first argument is the keyword `All`, otherwise only on the `AllMoving` or `Flapped` appendage identified by the label string or integer N_L.

The weights W_{ID} (depth), W_{IH} (heading), and W_{IR} (roll) are discussed in section 6.4. If the first argument is `All`, the second can be one of:

`Reset`. All weights are reset to their initial value, section 6.4.

`One`. All weights are set to a value of 1.0.

`Zero`. All weights are set to a value of 0.0.

For a specific appendage, arguments W_{ID} and W_{IH} are mandatory, and W_{IR} is optional. They must be in the range $-2 \leq W_{ID,H,R} \leq 5$. Multiple `WeightDCI` records can be used to reset weights for different appendages.

The global form of this record, '`WeightDCI All ...`' (only one of which is permitted per run), overrides weight values on individual appendages and clears the associated flags. Weights on specific appendages can then be reset individually. For example, to turn on all the appendages to indirect control commands except those numbered two and three, we can input:

```
WeightDCI All One
WeightDCI 2  0 0 0
WeightDCI 3  0 0 0
```

`WeightDCI` records should precede indirect `Delta` commands to which they apply.

Information on the initial status of some of the above quantities, notably the vehicle reference and indirect control weights, is output in the 'Configuration Summary' section at the beginning of file *Root*`.spr`. Most changes that result from resetting these and other parameters are recorded subsequently on the file as they occur.

20 Root.sim Level 2 Coefficnt Input

This option uses a series of internally generated static load calculations to estimate some linear hydrodynamic coefficients and stability and control indices. At present, coefficients are obtained using three-point fit code adapted, with minor modifications, from the `DERIVS` program [7]. The user is cautioned that the algorithms assume the vehicle has port-starboard symmetry, and that many of the coefficients will be incorrect if this is not the case. More general algorithms providing the higher-order coefficients and cross-coefficients required for a coefficient based simulation model will be developed for future builds of `DSSP21`.

The following table summarizes the keyword records for level 2 `Coefficnt` input:

Keyword	Function
`CB`	redefine CB location
`CG`	redefine CG location
`Del`	control derivative definition
`Delta`	control derivative definition
`DeltaLim`	reset control appendage deflection limits
`Derivs`	do a `DERIVS`-type calculation
`Length`	reference length for coefficients
`Output`	define output for `sim`*nnn*`.txt`
`Pop`	force termination of input — see section 5.4
`PReverse`	do plane reversal estimates
`Reference`	redefine reference origin and axes
`Rho`	redefine fluid density
`Text`	output run information or other descriptive text
`Ukt`	set total velocity in kt
`Ums`	set total velocity in m/s

20.1 Coefficient Definition Records: Del, Delta, Derivs, Length, Output, PReverse, Ukt, and Ums

These records control various aspects of coefficient and stability index [23] estimation. The coefficients are nondimensionalized in the standard form [24], and are output, multiplied by 1000, on *Root*`.spr`.

`Del`

Level 3 input that defines the control deflection derivative calculations will follow. The records that constitute this input are discussed in section 20.2.

`Delta`

As `Del`, above.

`Derivs`

This record is redundant since the `DERIVS` code three-point fit method is currently the only option for estimating coefficients (and is the default).

`Length` ℓ

Define reference length ℓ for nondimensionalizing the coefficients. It denotes vehicle length in the standard submarine equations of motion [24]. In `DSSP21`, ℓ defaults to hull length if there is only one hull present; otherwise (or to override the default) a `Length` record must be provided.

`Output` *keyword*

The keyword argument indicates the type of output to be put onto `sim`*nnn*`.txt`. If it is `Coef` or `Coefficnt`, the output is a list of the total vehicle coefficients multiplied by 1000. If it is `Root`, the output is a table of roots of the vertical stability equation for zero to critical speed. (As presented in this table, the roots may be discontinuous with speed.) Otherwise, or if the `Output` record is absent, `sim`*nnn*`.txt` will not be saved.

`PReverse` x_δ , [y_δ , z_δ]

This record initiates a small-perturbation plane reversal speed estimate [23] for a control surface located at $(x_\delta, y_\delta, z_\delta)$ in vehicle axes. Optional arguments y_δ and z_δ default to zero.

`Ukt` U

Define total velocity $U = \sqrt{u^2 + v^2 + w^2}$ for nondimensionalizing the coefficients, in kt; the default value is 1.944 kt (1 m/s). It is only necessary to set this parameter if there is a significant Reynolds number effect in the static load calculations.

`Ums` U

Define total velocity $U = \sqrt{u^2 + v^2 + w^2}$ for nondimensionalizing the coefficients, in m/s; the default value is 1 m/s. It is only necessary to set this parameter if there is a significant Reynolds number effect in the static load calculations.

20.2 Control Deflection Derivative Level 3 Input

Following a `Del` or `Delta` record, a sequence of records is required to define the control deflection derivatives to be estimated. The associated keywords are: `Down`, `Port`, `RollNeg`, `RollPos`, `Stbd`, `Starboard`, `Up`, and `WeightDCI`. A record beginning with any other keyword terminates the sequence.

Control deflection derivatives are associated with one of the indirect commands, the sense of which is immaterial in this context. None take an argument. Thus:

`Down` or `Up` results in calculation of vertical plane derivatives,

`Port`, `Stbd`, or `Starboard` results in calculation of horizontal plane derivatives, and

`RollNeg` or `RollPos` results in calculation of roll command derivatives.

`WeightDCI` records are included in this sequence to determine which appendages the above commands apply to, and how the appendages are weighted. They follow the syntax of section 19.4.

An example of control deflection derivative input is included in the *Root*`.sim` listing in annex E.

20.3 CB, CG, DeltaLim, Reference, Rho, and Text Records

These records can be input for most of the calculations and simulations on *Root*`.sim`; they redefine basic vehicle and simulation parameters that then remain unchanged until redefined in a subsequent run. The syntax and caveats applying to `DeltaLim`, `Reference`, and `Rho` are as presented in section 19.4. Hydrodynamic coefficients and other quantities are estimated with respect to the redefined reference. A `Text` record is simply output to *Root*`.spr` when it is encountered.

`CB` x_B , y_B , z_B

Redefines the three components of **CB** in vehicle axes.

`CG` x_G , y_G , z_G

Redefines the three components of **CG** in vehicle axes.

21 Root.sim Level 2 Captive Input

The aim of a `Captive` simulation is to estimate the loads seen by an internal model balance during a captive forced-motion experiment. A file tabulating some combination of velocity, Euler angle, and control deflection time histories is required to define the motion; but of these quantities, only values of time and forward speed u are mandatory. Rates of change of the tabulated data are estimated with a three-point quadratic fit. Since captive models are rarely exactly neutrally buoyant, and furthermore may have an atypical mass distribution compared with free-swimming vehicles, the user can input model mass and moments of inertia rather than rely on the default estimates.

The time history file format is consistent with `Free` simulation `sim`*nnn*`.txt` output, so a `Captive` simulation can be run after a `Free` one using the `sim`*nnn*`.txt` file previously created. However, unless the inertial properties of the model are changed or the hydrostatic loads are omitted, this is not a particularly useful exercise since the result is only the propulsor-equivalent loads seen by the notional balance together with some transients arising from imprecision in the estimation algorithms. The principal purpose of the captive simulation option is to estimate loads for manoeuvers such as arcs and combinations of harmonic motion using time history files generated by other software.

Having both the rotational velocities (with default or input initial Euler angles) and the Euler angles themselves constitutes redundant information because the velocities and angles can be derived from each other. Estimating Euler angles from the velocities requires integration and is susceptible to drift errors, so the angles themselves are used

as the primary source of data if they are (all three) available. If both quantities are present, a fatal error, '`TIMEHISTRY file P data is inconsistent`' (or '...`Q`...', or '...`R`...'), is generated if tabulated and estimated values do not agree. Without all three Euler angles on the time history file, the velocities, any of which defaults to zero if it is not present, are used to estimate the angles, ignoring any tabulated values.

The velocities, Euler angles, control deflections, and estimated loads are output to a `sim`*nnn*`.txt` file; estimated accelerations are output to an `aux`*nnn*`.txt` file.

The following table summarizes the keyword records for level 2 captive run input:

Keyword	Function
`Del`	set control appendage deflection default(s)
`Delta`	set control appendage deflection default(s)
`DeltaLim`	reset control appendage deflection limits
`Euler`	define initial Euler angles
`Iterate`	parameters for iteration of Euler angle estimates
`Mass`	input model mass
`MomInertia`	input model moments of inertia
`NoHStat`	omit hydrostatic loads
`Pop`	force termination of input — see section 5.4
`PQRtol`	tolerance for derivation of (p, q, r) from Euler angles
`Reference`	redefine reference origin and axes
`Rho`	redefine fluid density
`Text`	output run information or other descriptive text
`TimeHistry`	identify the time history file
`WeightDCI`	reset indirect control deflection weights

21.1 DeltaLim, Reference, Rho, Text, and WeightDCI Records

These records can be input for most of the calculations and simulations on *Root*`.sim`; they redefine basic vehicle and simulation parameters that then remain unchanged until redefined in a subsequent run. The syntax and caveats applying to `DeltaLim`, `Reference`, `Rho`, and `WeightDCI` are as presented in section 19.4. Motions on the time history file are taken to be motions of the redefined reference, and these should therefore be consistent with each other. Loads are calculated with respect to the redefined reference. A `Text` record is simply output to *Root*`.spr` when it is encountered.

21.2 Del, Delta, and TimeHistry Records

`Del` *Delta arguments*: see section 14

This sets default values for the one or more control deflections identified in the argument string. Any of the direct, quasi-direct, indirect, or global `Delta`

arguments discussed in section 14 can be used on this record. Multiple `Del` records can be present so long as a command conflict, section 14.5, is avoided.

In the absence of a `Del` command, the default for a control deflection is zero. If the time history file described below contains deflection data for the same control appendage, the time history data are used and the default is ignored.

`Delta` *Delta arguments*: see section 14

As `Del`, above.

`TimeHistry` *filename*

The mandatory argument is the name of the time history file; its length, including extension, cannot exceed twelve characters.

The time history file comprises a header record with a case-insensitive name for each of the data columns, and a series of data records; it can also include comment or `Text` records, which are respectively ignored or output to *Root*`.spr`. The first data column must be time, monotonically increasing, identified on the header by `Time`. The only other required data column is u, identified by `U`.

Columns of v, w, p, q, r, ϕ, θ, and ψ, identified by `V`, `W`, `P`, `Q`, `R`, `Phi`, `The`, and `Psi` are optional, as are control deflections, identified by `Del`N_L for appendage N_L. Other than `Time`, the columns can be in any order, need not be aligned, and those identified by names not mentioned above are ignored.

Translational velocities must be in m/s, rotational velocities in deg/s, and angles and control deflections in degrees.

There is insufficient room to illustrate a typical time history file here, but the following trivial example illustrates some of the key points above:

```
text this is tiny

time  w  u  del4 THE          " Since not all three Euler angles are
                              " input, the <The> column is ignored.
1   0   2    6      1.2       " In addition, p, q, and r default to
2  .1   3    6      1.25      " zero, so the Euler angles used are
3  .11  4    6.2  1.27        " all zero.
5  .12  4    6.2  1.28        "
6  .09  4.1  6.1   1.282      " There will be, however, loads in six
7  .02  3.9  5.5   1.285      " degrees of freedom because del004
9 -.06  3.4  4.8  1.1         " is a vertical rudder.
```

21.3 Euler, Iterate, Mass, MomInertia, NoHStat, and PQRtol Records

These records control various aspects of load estimation.

`Euler` $\phi_\bullet$, $\theta_\bullet$, $[\psi_\bullet]$

If the Euler angles of the model are not tabulated on the time history file, they are estimated from rotational velocities using initial values that default to zero. This record is required to define non-zero initial values; $\psi_\bullet$ is optional. The Euler angle calculations are coupled, and the initial values of ϕ and θ are not simply added to their resulting time histories. This means that $\phi_\bullet$ and $\theta_\bullet$ must be correct in order to correctly generate the dynamics of the simulation.

There is no reason to provide this record if angle time histories are present. If that is done, a fatal error results if the input and time history initial angles are inconsistent.

`Iterate` N_{iter} , ϵ_{iter}

An iterative trapezoidal integration is used to derive Euler angles from rotational velocities. The arguments on this record are the maximum number of iterations N_{iter} (default = 30) and the tolerance for convergence ϵ_{iter} (default= 0.0001). Conditions under which nonconvergence, indicated by the fatal error '`Euler angle estimates did not converge - p,q,r resolution or conditioning problem`', will occur have not been fully identified, and some experimentation with these parameters will be required if the error is encountered.

`Mass` m

The argument m, in kg, replaces the default model mass estimated by a neutral buoyancy calculation. Mass represented by m is independent of the density ρ used elsewhere in the program.

`MomInertia`

This record is followed by a one or two record data block containing moments of inertia I_{xx}, I_{yy}, I_{zz} and, optionally, cross products of inertia I_{xy}, I_{xz}, I_{yz}, all in kg·m^2, that will replace the default model values estimated from neutral buoyancy and uniform model density. The data block format is:

$$\begin{matrix} I_x & I_y & I_z \\ [\, I_{xy} & I_{xz} & I_{yz} \,] \end{matrix}$$

If the record containing cross products is omitted, they are set to zero. Input values of all these quantities are independent of the density ρ used elsewhere in the program. They are assumed to be consistent with the reference origin specified for the simulation.

`NoHStat`

By default, hydrostatic loads (e.g., $(W - B)\sin\phi\cos\theta$ in the Y equation of motion [24]) are included in a `Captive` simulation. They are omitted if this record is present.

`PQRtol` ϵ_ω

This record overrides the default tolerance used to check for consistency between (p, q, r) time histories and the values estimated from Euler angle time histories. The default value of ϵ_ω is 0.01, which the program interprets as deg/s to compare angular velocities and as deg/s^2 to compare angular accelerations. If inconsistency is found a fatal error of the form '`TIMEHISTRY file P/Q/R data is inconsistent`' is generated. While increasing ϵ_ω may remove the error in marginal cases, it is preferable, and generally necessary, to rigorously eliminate inconsistencies from the time history file. If the Euler angle values are unreliable, the simplest way to ignore them and use the estimated values instead is to rename one of the Euler angle columns in the time history file with an unrecognized string, such as '`foo`'.

22 Root.sim Level 2 Free Input

The `Free` keyword record initiates a manoeuvering simulation run for a free-swimming vehicle. The equations of motion and various auxiliary differential equations (for control dynamics, etc.) are integrated over time using an order (3,4) Runge-Kutta method with adaptive timestep for local error control [8]. Output may be obtained at each timestep, or at a regular interval by means of interpolation [9]. Even while interpolating, actual timesteps are output to the *Root*`.log` file for diagnostic purposes.

At the end of each manoeuvering simulation, estimates of computer time for the simulation and for additional overhead (usually negligible), and the ratio of simulated time to elapsed computer time are output to *Root*`.spr`. The ratio, which is generally of the order of 10^3, will help determine whether a particular manoeuver or selected options are unduly slowing down the calculation. For example, small time steps when blowing a ballast tank from more than one bottle group may slow the calculation by nearly an order of magnitude. (One way to alleviate the problem is to increase minimum step size and tolerance in the integration so long as convergence tests show that an acceptable level of precision is maintained.)

Initially, the vehicle is moving in a horizontal straight line parallel with the earth-fixed x_0 axis (section 3.4), in level trim and neutrally buoyant; initial depth and speed are defined by the user. These conditions are maintained until a command is given that causes them to change.

The (x_0,y_0,z_0) axes can be equivalenced to an initial compass heading and chart axis location. In the current build of the program, compass heading is optionally used for autopilot control, and chart location is not required at all; however, both are output. In future builds they might be used for relating manoeuvers to a specific bottom topography, a current field, etc.

`Free` input at level 2 in *Root*`.sim` consists of initialization records, which provide data for defining or initializing various run and vehicle parameters, and command records,

which specify a control action to take place during the simulation. Command records are sorted into bundles of one or more simultaneous commands. Time integration from one command bundle to the next is called a 'sprint'. The simulation is terminated when the last sprint is complete. Null commands `Start` and `Stop` are provided to start and stop the simulation without any control action; clearly, stopping with a control action is unproductive since the action will not have any effect.

Initialization records are executed as they are encountered, and have the typical form:

keyword , [argument(s)]

whereas command records are distinguished by having a floating-point numerical value, *time*, as the first word in the record:

time , keyword , [argument(s)]

and are executed at the indicated time during the simulation. They currently specify control deflections, changes in propulsor RPM, autopilot control, ballast blowing or venting, and compartment flooding. An error in a command record may not be flagged by the program until the sprint including the command is executed.

The same keyword may have somewhat different interpretations in an initialization record and in a command record because the context is different. For example, as we shall see in sections 22.1 and 22.4,

```
Delta #UpperRudder 2
```

initializes the appendage named `#UpperRudder` at a setting of +2.0 degrees before the simulation is started, whereas

```
100.0 Delta #UpperRudder 2
```

commands the appendage to commence deflecting to an angle of +2.0 degrees at time $t = 100$ s during the simulation.

The order of initialization records is significant in one case: `WeightDCI` records should precede indirect `Delta` initialization, as was noted in section 19.4. Otherwise, the order of records, and the mixing of initialization and command records, is immaterial. Allowing this mixing facilitates input trial-and-error in simulating a manoeuver since new commands being tested can be placed at the end of the input where they are easily modified without concern whether they are correctly placed in sequence of time. However, it is recommended that the input be reorganized into a rational sequence for archiving once a manoeuver is satisfactory.

The following are the initialization record keywords for level 2 free run input:

Keyword	Function
`CB`	redefine CB location
`CG`	redefine CG location

`Del`	initialize control appendage deflection(s)
`Delta`	initialize control appendage deflection(s)
`DeltaLim`	reset control appendage deflection limits
`Depth`	initialize depth in m
`GimbaLock`	avoid or allow gimbal lock
`Heading`	initialize compass heading
`Interpl`	set output interpolation parameters
`Isothermal`	use isothermal ballast blowing
`NoMass`	do not update inertial properties for a mass change
`Origin`	initialize position in chart axes
`Pop`	force termination of input — see section 5.4
`Reference`	redefine reference origin and axes
`Rho`	redefine fluid density
`RPM`	initialize propulsor RPM(s)
`SVDecomp`	use SVD algorithm to invert the $(\mathbf{I} + \mathbf{A})$ matrix
`Target`	define target point
`Text`	output run information or other descriptive text
`TimeStep`	set lower and upper timestep limits
`Tolerance`	set integration tolerance
`Ukt`	initialize axial velocity in kt
`Ums`	initialize axial velocity in m/s
`WeightDCI`	reset indirect control deflection weights

The following are the command record keywords:

Keyword	Function
`AutoDepth`	command depth autopilot
`AutoHead`	command heading autopilot
`AutoProp`	command propulsion autopilot
`AutoRoll`	command roll autopilot
`Blow`	blow MBT(s)
`Del`	command control appendage deflection(s)
`Delta`	command control appendage deflection(s)
`Dummy`	a null command
`Flood`	flood a WTC
`Prop`	command propulsor RPM
`Propulsor`	command propulsor RPM
`Start`	start the simulation
`Stop`	stop the simulation
`Vent`	vent MBT(s)

A text record is simply output to *Root*`.spr` when it is encountered.

22.1 Vehicle Initialization Records: CB, CG, Del, Delta, DeltaLim, Reference, and Target

`CB`, `CG`, `DeltaLim`, and `Reference` records can be input for most of the calculations on *Root*`.sim`; their syntax is as presented in sections 4.2 and 19.4. Note the caveats given in sections 3.4 and 4.5 regarding the reference for a free manoeuvering simulation.

`Del` and `Delta` are used to define initial control deflection(s) at the start of the simulation. These deflections are taken into account in establishing the initial condition of level flight with zero trim. The standard `Delta` syntax is used.

`Del` Delta arguments: see section 14

Any of the direct, quasi-direct, indirect, or global `Delta` arguments discussed in section 14 can be used on this record. Multiple `Del` records can be present so long as a command conflict, section 14.5, is avoided.

`Delta` Delta arguments: see section 14

As `Del`, above.

`Target` x_T , y_T , z_T

The point defined by (x_T, y_T, z_T) in vehicle axes replaces the reference point as the target tracked in chart axes on files *Root*`.spr` and `aux`*nnn*`.txt`.; see section 3.4.

22.2 Condition Initialization Records: Depth, Heading, Origin, Rho, RPM, Ukt, Ums, and WeightDCI

`Rho` and `WeightDCI` records can be input for most of the calculations on *Root*`.sim`; their syntax is as presented in section 19.4.

`Depth` $z_{0\bullet}$

Defines initial depth, $z_{0\bullet}$, in m, it must not be negative.

By default, $z_{0\bullet} = 0$; this must be changed to an appropriate value if the simulation includes blowing, venting, or flooding.

`Heading` *string*

Defines initial compass heading using one of the argument string formats described in section 15. By default it is zero.

`Origin` $x_{e\bullet}$, $y_{e\bullet}$

Defines initial chart axis x_e and y_e coordinates of the target point in m; the initial z_e coordinate is defined by `Depth` ($z_{e\bullet} = z_{0\bullet}$). By default, $x_{e\bullet} = y_{e\bullet} = 0$.

`RPM` $[n_\bullet]$

Defines initial values of RPM for the propulsors, and thus defines, indirectly, the initial speed. If there is only one propulsor, $n_\bullet$ can be on the same record; otherwise, a value of $n_\bullet$ for each propulsor is given in a following data block, three to a record. There can be only one initialization record with the keyword `RPM`, `Ukt`, or `Ums`.

`Ukt` $U_\bullet$

Defines the initial forward speed in kt, and thus defines, indirectly, initial values of RPM for the propulsors. There can be only one initialization record with the keyword `Ukt`, `Ums`, or `RPM`.

`Ums` $U_\bullet$

Defines the initial forward speed in m/s, and thus defines, indirectly, initial values of RPM for the propulsors. There can be only one initialization record with the keyword `Ums`, `RPM`, or `Ukt`.

22.3 Simulation Initialization Records: GimbaLock, Interpl, Isothermal, NoMass, SVDecomp, TimeStep, and Tolerance

`GimbaLock` [*keyword*]

By default the gimbal lock singularity when the vehicle is pitched exactly ± 90 degrees [25] is avoided by using a quaternion representation of the Euler angles; the angles themselves are derived at each timestep for output. Euler angles can be calculated explicitly by including this record with the keyword argument `Euler`. The alternative argument, `Quaternion`, is redundant in this version of the program.

`Interpl` Δt , [*keyword*]

Defines an output interpolation interval Δt in seconds. Output in *Root*`.spr` and `sim`*nnn*`.txt` is given at each time $t = k\Delta t$, where k is integer, although internal calculations are still done with the adaptive timestep for error control. The optional keyword second argument indicates a fourth-order interpolant to use; it is `TypeZero` or `TypeOne` for, respectively, C^0 or C^1 continuity [8,9]. The former, which is the default, is computationally more economical but may perform less well in some circumstances. In the absence of this record, calculations are output at the adaptive timesteps.

Only the extended state variable set is interpolated; other quantities will normally have a value calculated at the last integration timestep. The output file principally affected is `aux`*nnn*`.txt`, which tabulates a number of intermediate and derived quantities. Those quantities for which interpolation results in an unacceptable loss of precision on output are recalculated at the interpolation timesteps.

`Isothermal`

This record forces the simple isothermal model for ballast tank blowing; the default is an adiabatic blow.

`NoMass`

A change in the mass of the submarine, during a ballast blow for example, is by default reflected in an adjustment of all its inertial properties, and requires recalculation and inversion of the inertial plus added mass matrix, $(\mathbf{I}+\mathbf{A})$, at each time step following the change. The `NoMass` record is an option to reduce the associated overhead by the approximation of keeping the inertial properties constant while simply adding the resulting additional forces and moments to the equations of motion. This approximation appears to be acceptable for simulating the blow of a typical reserve of buoyancy of ten percent on a submarine, but for larger mass changes such as may occur in smaller vehicles, the validity of the approximation should be assessed before accepting it.

`SVDecomp` $[\epsilon_{\text{svd}}]$

To try to circumvent ill-conditioning of the inertial plus added mass matrix when the reference is moved away from the vicinity of the CB and CG (sections 3.4 and 4.5), a singular value decomposition (SVD) matrix inversion algorithm from Press *et al* [26] is invoked with this record. The optional argument ϵ_{svd} is the threshold for discarding elements of the matrix [26]; it has a default value of 10^{-6}. Note that in SVD some accuracy is lost. If more than one element is discarded, accuracy will likely deteriorate to an unacceptable level, but discarding just one may not be sufficient to complete the matrix inversion. Diagnostic output is provided in the *Root*`.log` file to guide the selection of ϵ_{svd}.

Unless the user has overpowering reasons to shift the reference, it is preferable to select a target point, section 22.1, to track a particular trajectory.

`TimeStep` h_{min} , h_{max}

Defines limits, in seconds, on the internal adaptive timestep h. Both should be positive and $h_{min} \leq h_{max}$, but if either is zero the corresponding default is used; e.g., to increase maximum step size to 5 seconds, use '`TimeStep 0 5`'. The default for h_{min} is a small value based on the tolerance, ϵ, and computer roundoff error; the default for h_{max} is 2 seconds. New timestep limits set by this record will apply to subsequent simulations in the same *Root*`.sim` file.

As noted, h_{min} is related to ϵ; if it is increased, ϵ may have to be increased. Current experience adjusting these parameters suggests that their input values should be of the same order. If h_{min} is increased relative to ϵ sufficiently that the requested tolerance cannot be achieved, the program will report a fatal error: '`Unexpected IND = -3 from DDNORK ...`'. (But note that there may be other reasons for failing to meet tolerance, see section 3.3.)

A similar error '`... IND = -2 ...`' will result if $h_{min} > h_{max}$.

`Tolerance` ϵ

This record defines a value for integration tolerance [8,9]; it must not be negative. The default value, unless redefined by a previous free run in the same *Root*`.sim` file, is 0.001. See `TimeStep` for the relationship between tolerance and minimum timestep.

22.4 Command Records: AutoDepth, AutoHead, AutoProp, AutoRoll, Blow, Del, Delta, Dummy, Flood, Prop, Propulsor, Start, Stop, and Vent

Recall that these records have the format:

t , *keyword* , [*argument(s)*]

where t, is the absolute value, in seconds, of time at which the command will be issued. Time t is absolute because the simulation is not constrained to start at $t = 0$, but is started at the time of the earliest command. Similarly, it is stopped at the time of the latest command.

In free runs, control deflections, RPM, etc., are dynamic quantities, so the value specified in a command is not attained instantly. The parameters defining control and propulsor dynamics are discussed in sections 6.5 and 8.3 respectively.

The keyword and arguments in a command record are interpreted as follows:

`AutoDepth` [*keyword*], [*keyword*], [z_{0c}]

The first optional keyword determines control options (see section 10): it may be `Depth`, to use depth control only; `Pitch`, to use pitch control only; `TwoParam`, to use both; or `Stop`, to stop depth control. The default is `TwoParam`. Following `Stop`, associated control surfaces are returned to their initial deflection in the simulation unless they are acting under other indirect commands.

The second optional keyword determines autopilot type: it may be `PhaseLead` or `PID`; these options and the default settings are described in section 10.1.

The optional third argument is commanded depth in m. If this argument is omitted, z_{0c} is made equal to the current depth; a command record '*t* `AutoDepth`' is interpreted as 'maintain current depth with autopilot'.

`AutoHead` [*keyword*], [*keyword*], [*string*]

The first optional keyword argument determines the direction to turn into the commanded heading: it may be `Port`/`Stbd`/`Starboard`, requiring the turn to be made in the specified direction; `Shortest`, meaning 'take whichever direction is shortest'; or `Stop`, to stop heading control. The default is `Shortest`.

Following `Stop`, associated control surfaces are returned to their initial deflection in the simulation unless they are acting under other indirect commands.

The second optional keyword determines autopilot type, see `AutoDepth`.

The optional third argument is commanded heading ψ_c expressed in one of the argument string formats described in section 15. By default ψ_c is made equal to the current heading; i.e., 't `AutoHead`' is interpreted as 'maintain current heading with autopilot'.

`AutoProp` *keyword*, [*keyword*], [Q_c]

The first keyword argument is mandatory; it determines the propulsion control parameter. In the absence of autopilot control, the propulsion model uses constant RPM. With autopilot control, RPM is adjusted according to demanded power, speed, or thrust; the choice is indicated by this keyword being `Power`, `Ukt` or `Ums`, or `Thrust`, respectively. Alternatively, if the keyword is `Stop`, propulsion control is stopped and RPM is maintained at the current value until changed by another command.

The second optional keyword determines autopilot type, see `AutoDepth`.

The optional third argument is a demand value for the control parameter. If the first keyword is `Power`, Q_c is power in W; if `Ukt`, it is speed in kt; if `Ums`, it is speed in m/s; and if `Thrust`, it is thrust in N. By default, Q_c is made equal to the current value of power, speed, or thrust according to the first keyword. Thus a command record 't `AutoProp Power`' is interpreted as 'maintain current propulsion power under autopilot control'.

If specifying a demand value for power or thrust, the user should recall that the resistance and propulsion models in `DSSP21` are quite general, and may not predict these quantities very accurately for a particular submarine. Power and thrust are output to the `aux`*nnn*`.txt` file. It is advisable to establish speed-power and speed-thrust relationships with a few simple simulations before inputting specific demand values for these quantities.

`AutoRoll` [*keyword*], [*keyword*], [ϕ_c]

The optional first keyword argument may be `Shortest`, which is redundant and should be omitted, or `Stop`. Following `Stop`, associated control surfaces are set to zero unless they are acting under other indirect commands.

The second optional keyword determines autopilot type, see `AutoDepth`.

The optional third argument is commanded roll angle, ϕ_c, in degrees. By default, ϕ_c is zero. Thus a command record 't `AutoRoll`' is interpreted as '(try to) maintain zero roll angle under autopilot control' — however, most underwater vehicles will have insufficient roll control authority to do so unless roll control is integral to their design.

`Blow` *string*

The argument string is a combination of the bottle group and ballast tank arguments discussed in section 16. A ballast tank can only be referred to once in a bundle of simultaneous `Blow` commands.

`Del` *Label String* or N_L or *keyword* , δ

Any of the direct, quasi-direct, indirect, or global `Delta` arguments discussed in section 14 can be used on this record. Multiple simultaneous `Del` and `Delta` commands can be input so long as a command conflict, section 14.5, is avoided.

`Delta` *Label String* or N_L or *keyword* , δ

As `Del`, above.

`Dummy`

The `Dummy` command generates no command action. However, it can be used as a time marker since, like all command records, it stops and restarts the integration.

`Flood` *string*

The argument string is discussed in section 17; it identifies the WTC to be flooded and defines the flooding breach. A WTC can be referred to only once in a bundle of simultaneous `Flood` commands.

`Prop` [*Label String* or N_P] , n

If the first argument is omitted, and there is only one propulsor, then that propulsor is commanded to n RPM, but if there is more than one propulsor, a fatal error is generated. However, if the first argument is present, the propulsor identified by the label string or index N_P is commanded to n RPM.

`Propulsor` [*Label String* or N_P] , n

As `Prop`, above.

`Start`

The `Start` command is currently implemented similarly to `Dummy` in that it generates no commanded action; however, it cannot be later in time than any other command. Because of the way `DSSP21` determines the bounds of the simulation — from the earliest command to the latest — it provides an easily recognizable way of defining the start time.

`Stop`

The `Stop` command is currently implemented similarly to `Dummy` in that it generates no commanded action; however, it cannot be earlier in time than any other command. Because of the way `DSSP21` determines the bounds of the simulation — from the earliest command to the latest — it provides an easily recognizable way of defining the stop time.

`Vent` *string*

The argument string is a combination of the ballast tank arguments discussed in section 16. A ballast tank can only be referred to once in a bundle of simultaneous `Vent` commands.

23 Running the Program, and Interactive Dialogues

`DSSP21` has been written to run from the input files with virtually no user intervention. Unless the *Root* character string that is required to identify input and output files is detected as a command line parameter, the user is asked to supply it at the start of the program:

```
Enter file name root (default = dssp21):
```

A string up to eight characters long is expected. Enter/Return indicates the default '`dssp21`'. Not finding the geometry input file *Root*`.geo` generates a fatal error.

The methods for calculating hull residual drag, section 5.3, are likely inaccurate for truncated hulls for which the base is large. Since there is not a good criterion for largeness in this context, the program asks:

```
Calculate residual drag for truncated hull ... (Y/[N])?
(It may be incorrect)
```

The default sets the residual drag to zero. The message is suppressed by supplying a value for `ResiDragR`, section 5.3.

An unlikely case requiring user intervention occurs with the `DSSPtwo` propulsor model. This model requires a scaling length equivalent to LBP, and the following message is generated for a vehicle with no hull components:

```
(KTKQ02) Enter scaling length (LBP)
```

The value entered must be positive.

In addition, there are a few conditions that result in a warning with pause, section 3.3, for which the user is asked whether or not to continue with program execution. They mostly occur during geometry and systems setup in the first phase of execution.

24 Output Files

For the most part, output from `DSSP21` is self-explanatory. It is also too voluminous to illustrate here. For error checking, the *Root*`.log` file echoes all input from *Root*`.geo` and *Root*`.sim`, reports a number of intermediate calculation steps, and outputs all error

messages. Most errors in input can be identified by comparing the error message and the last echoed line of input. Those in a `Free` simulation command, however, will not be flagged until the command is executed.

While a *Root*`.geo` file is being processed, input data, intermediate results, and hydrodynamic and other characteristics are output to *Root*`.gpr`. The vehicle components and systems therefore appear in the same order as on the input file. At the end, there is a summary of propulsor initialization, a table of overall vehicle properties, tables of the added and inertial masses and moments of inertia, if calculated, and a table of autopilot parameters. The level of detail in *Root*`.gpr` has been useful while the code is under development, but is probably excessive for most purposes. Some of it may be made optional in later builds.

Root`.spr` presents a summary of conditions and results for each of the runs called for in *Root*`.sim`. Its purpose is to confirm that the run parameters and other conditions were properly set up, especially if there are anomalies in the results. The principal output of calculations and simulations is to the `sim`*nnn*`.txt` and `aux`*nnn*`.txt` files, and is in simple tabular form for importing into spreadsheet, graphical, or other post-processing software.

Root`.spr` starts with a vehicle configuration summary that includes a list of components by number and label, indirect deflection control weights, a table of appendages associated with various quasi-direct and indirect commands, and reference axes information. For each run, the number *nnn* of the corresponding `sim`*nnn*`.txt` and `aux`*nnn*`.txt` files is given. `Static` run output merely lists values of the run parameters and the number of iterations if a `Loop` is executed. `Coefficnt` run output lists all the coefficients and stability indices calculated. `Free` simulation output includes run initialization data and the target trajectory time history in chart axes, i.e., (t, x_T, y_T, z_T), at each time, or interpolation, step.

A `sim`*nnn*`.txt` file tabulates loads for a `Static` run, coefficients (if the `Output` option is selected, section 20.1) for a `Coefficnt` run, and load time histories for a `Captive` run. For a `Free` run it tabulates time histories the extended state variable data: time, speed, (x_0, y_0, z_0) coordinates, Euler angles, compass heading, translational and rotational speeds in boat axes, control deflections, and RPM. An `aux`*nnn*`.txt` file tabulates time histories of some useful auxiliary data: chart coordinates, power, thrust, autopilot error and control signals, bottle group pressures and temperatures, water displaced from ballast tanks and floodwater mass.

It is noted above that some data in the `aux`*nnn*`.txt` files are approximated during interpolation by their value at the last internal timestep. Although the program recalculates those quantities for which the approximation would lead to an unacceptable loss of precision, it is not done for those that are considered to be primarily for qualitative or diagnostic purposes, e.g., thrust, power, autopilot signals, and changes in vehicle mass.

25 Concluding Remarks

This document is a user guide to build 061102 of the underwater vehicle manoeuvering simulation code DSSP21, with particular emphasis on input preparation. It is aimed at both experienced and new users.

Some features of the DSSP21 current build are still under review. Validation of many of the algorithms is on-going and improvements will continue to be made as new data become available. Nevertheless, the current build is amply capable to be documented in this guide for use in hydrodynamic studies and as a benchmark for further development.

References

For proprietary and other reasons, references denoted Limited Distribution or Private Communication are not for public release.

1. Mackay, M. (1999). DSSP20 (Beta Edition) User Guide to the Preprocessing Modules. (DREA TM 1999–108). Defence Research Establishment Atlantic.

2. Mackay, M. (1999). DSSP20 (Beta Edition) User Guide to the Simulation Modules. (DREA TM 1999–109). Defence Research Establishment Atlantic.

3. Mackay, M. (1995). Estimation of the Force due to a Submarine Sail or Similar Appendage. In: *Third Canadian Marine Hydrodynamics and Structures Conference*. Halifax: Technical University of Nova Scotia.

4. Mackay, M. (1995). A Review of Semiempirical Methods for Predicting Appendage Forces. (DREA Report 95/102). Defence Research Establishment Atlantic. Limited Distribution.

5. Mackay, M. (2003). The Standard Submarine Model: A Survey of Static Hydrodynamic Experiments and Semiempirical Predictions. (DRDC Atlantic TR 2003-079). Defence R&D Canada – Atlantic.

6. Watt, G.D. (1988). Estimates for the Added Mass of a Multi-Component, Deeply Submerged Vehicle, Part I: Theory and program Description. (DREA TM 88/213). Defence Research Establishment Atlantic.

7. Watt, G.D. (1998). Estimating Underwater Vehicle Stability and Control Derivatives using ESAM and a Preliminary version of DSSP20. (DREA TM 98/224). Defence Research Establishment Atlantic. Limited Distribution.

8. Enright, W.H., Jackson, K.R., Nørsett, S.P., and Thomsen, P.G. (1986). Interpolants for Runge-Kutta Formulas. *ACM Transactions on Mathematical Software*, Vol. 12, No. 3. Association for Computing Machinery.

9. Enright, W.H., Jackson, K.R., Nørsett, S.P., and Thomsen, P.G. (1988). Effective Solution of Discontinuous IVPs Using a Runge-Kutta Formula Pair with Interpolants. *Applied Mathematics and Computation*, Vol. 27. Amsterdam: Elsevier.

10. Etkin, B. (1972). *Dynamics of Atmospheric Flight*, Second Edition. New York: Wiley.

11. Hooft, J.P. (1986). Hydrodynamic Forces on Tear-Drop Bodies. (MARIN Report No. 07659–1–MO). Maritime Research Institute Netherlands. Limited Distribution.

12. Mackay, M. (2006). Estimating the In-Plane Hydrodynamic Loads on a Submarine Hull at Incidence. (DRDC Atlantic TM 2006–020). Defence R&D Canada – Atlantic.

13. Mackay, M. (1992). DREA Submarine Simulation Program Version 0.2 (DSSP02) — Release Notes. (DREA TC 92/308). Defence Research Establishment Atlantic. Limited Distribution.

14. Watt, G.D. and Fournier, E.Y. (1995). Submarine Propulsion Testing in the IAR 9 m Wind Tunnel. In: *Third Canadian Marine Hydrodynamics and Structures Conference*. Halifax: Technical University of Nova Scotia.

15. van Lammeren, W.P.A., van Manen, J.D., and Oosterveld, M.W.C. (1969). The Wageningen B–Screw Series. *SNAME Transactions*, Vol. 77. New York: Society of Naval Architects and Marine Engineers.

16. Kennedy, J.L. (1998). Private Communication. Defence R&D Canada – Atlantic.

17. Hoerner, S.F. (1958). *Fluid-Dynamic Drag*. Midland Park NJ: published by the author.

18. Dempsey, E.M. (1977). Static Stability Characteristics of a Systematic Series of Stern Control Surfaces on a Body of Revolution. (DTNSRDC Report 77–0085). David Taylor Naval Ship Research and Development Center.

19. Mackay, M., Bohlmann, H.J., and Watt, G.D. (2002). Modeling Submarine Tailplane Efficiency. In *Challenges in Dynamics, System Identification, Control and Handling Qualities for Land, Air, Sea, and Space Vehicles*. (RTO–MP–095). Paris: NATO RTO.

20. Mackay, M. (2004). A Review of Submarine Out-of-Plane Normal Force and Pitching Moment. (DRDC Atlantic TM 2004-135). Defence R&D Canada – Atlantic.

21. El-Hawary, M.E. (1984). *Control System Engineering*. Reston VA: Reston Publishing.

22. Mackay, M. (1996). Submarine Turning Circle Experiments in a Towing Tank. In: *RINA Warship '96 Symposium: Naval Submarines 5*. London: Royal Institution of Naval Architects.

23. Mackay, M. (2001). Some Effects of Tailplane Efficiency on Submarine Stability and Manoeuvering. (DRDC Atlantic TM 2001-031). Defence R&D Canada – Atlantic.

24. Gertler, M. and Hagen, G.R. (1967). Standard Equations of Motion for Submarine Simulation. (NSRDC Report 2510). Naval Ship Research and Development Center.

25. Phillips, W.F., Hailey, C.E., and Gebert, G.A. (2000). A Review of Attitude Kinematics for Aircraft Flight Dynamics. (AIAA paper 2000–4302). In: *AIAA Modeling and Simulation Technologies Conference, Denver, CO*. American Institute of Aeronautics and Astronautics.

26. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992). *Numerical Recipes in Fortran*, Second Edition. Cambridge: CUP.

Annex A. Program Change List

DSSP20

DSSP20, the predecessor to DSSP21, had the same objective of predicting the hydrodynamic characteristics and manoeuverability of an underwater vehicle, but was implemented as a suite of interacting programs. References [1] and [2] are user guides for build 981009. Subsequent builds of DSSP20 (to 010830) improved and expanded a number of the hydrodynamic load algorithms, but the input specification described in these guides changed very little. Although the individual programs worked well, maintainability of the suite as a whole became more difficult as their complexity increased.

DSSP21

A number of DSSP21 builds previous to 061102 have been archived but not formally documented; only the principal prior benchmark, build 050401, is noted here in addition to the current build.

Build 050401

The individual programs in DSSP20 were combined into one, resulting in about fifty percent of the original code being rewritten. Most of the additional changes in this build addressed consistency and usability issues; the hydrodynamic load algorithms were largely unchanged. Changes included:

- Allowing a tab as an input delimiter.
- Eliminating questionable Fortran usage such as floating point equality and inequality testing.
- Addressing inconsistencies in angular rate input and output; now deg/s is used for all state rotations, and rad/s is used for control parameters.
- Modifying Hull discretization to allow arbitrary station spacing and numbers, within limits.
- More rational Propulsor input.
- Rationalizing input and estimation of **BG**, **CB**, and **CG**.
- Implementing sim*nnn*.txt and aux*nnn*.txt output for postprocessing.
- Implementing the DERIVS hydrodynamic coefficient estimation algorithms.
- Implementing the quaternion representation of Euler angles.

Build 061102

Additions implemented in build 061102 include:

- Autopilots, and compass heading and course commands.
- VRML 1 plotting output.
- Optional command line input of the `Root` string.
- Isothermal and adiabatic ballast blowing, and venting.
- Captive vehicle simulations.
- Out-of-plane load time history calculations.
- Bohlmann-Spreiter tailplane efficiency calculations.
- Watertight compartment flooding.
- Improved hull load calculations; provision for deck and keel corrections.

Annex B. Dictionary Lists

Dictionary lists are included in the output generated by the `Help` command, which always gives the most up-to-date listings. Keywords are upper case in the dictionary, but user input is not case-sensitive.

The compass heading dictionary list of build 061102 comprises:

```
NORTH      N          NBYE       NNE        NEBYN
NE         NEBYE      ENE        EBYN       EAST
E          EBYS       ESE        SEBYE      SE
SEBYS      SSE        SBYE       SOUTH      S
SBYW       SSW        SWBYS      SW         SWBYW
WSW        WBYS       WEST       W          WBYN
WNW        NWBYW      NW         NWBYN      NNW
NBYW
```

The main dictionary list of build 061102 comprises:

```
ACONCOEFF  ACROSPIN   ADDEDMASS  AFT        ALFA
ALL        ALLMOVING  ALPHA      AUTODEPTH  AUTOHEAD
AUTOPROP   AUTOROLL   AZIMUTH    BASEAREA   BETA
BG         BLOW       BOHLMANN   BOWPLANE   CAMBER
CAPTIVE    CB         CG         CLEAR      COEF
COEFFICNT  COLOUR     CXJ        CXZERO     DEBUG
DECK       DEFAULT    DEL        DELNONLIN  DELRATELIM
DELTA      DELTADYN   DELTALIM   DEMPSEY    DEPCONGAIN
DEPERRGAIN DEPTH      DERIVS     DIAMETER   DIAMRATIO
DOWN       DSSPTWO    DUMMY      EMERGENCY  ESAM
EULER      FAUXBLOW   FLAPGAP    FLAPPED    FLAPRODRAG
FLOOD      FOREPLANE  FORM       FORWARD    FREE
FWD        GIMBALOCK  HARMONIC   HASENDP    HEADING
HEDCONGAIN HEDERRGAIN HELP       HOERNER    HPAIR
HULFOM     HULFOR     HULL       HULLSEGMNT HULLSEP
INERTIAL   INITDELTAS INTERPL    IRREGULAR  ISENDP
ISOTHERMAL ITERATE    KEEL       KILL       KPARAM
KRIGING    LABEL      LAMBDA     LBOWPLANE  LEFT
LENGTH     LIFT       LIFTSEGMNT LOCATION   LOD
LOOKAHEAD  LOOP       LUMP       MAIN       MAPLE
MASS       MASSFLOW   MATLAB     MBT        MIDHULL
MLD        MOMINERTIA NEURALNET  NODEBUG    NOHSTAT
```

```
NOMASS      NONE        NORMAL      NOSE        OCCUPANCY
ONE         OOPCALC     OPEN        ORIGIN      OUTPUT
P           PHASELEAD   PHI         PID         PITCH
PITCHLIMIT  PLANFORM    PLOT        POD         POINTS
POP         PORT        POWER       PQRTOL      PRESSURE
PREVERSE    PROP        PROPULSOR   PSI         PWRERRGAIN
Q           QUATERNION  R           RADIUS      REFERENCE
REFLECTED   RELATIVE    RESET       RESIDRAGR   REVCONGAIN
REVDYNAM    REVLIMIT    REVRATELIM  RHO         RIGHT
ROLCONGAIN  ROLERRGAIN  ROLL        ROLLNEG     ROLLPOS
ROOT        ROTATED     ROUGHALLOW  RPM         RUDDER
SAIL        SAILPLANE   SAVE        SBOWPLANE   SHIPSTAB
SHORTEST    SOURCE      STAMP       STARBOARD   START
STATIC      STATION     STBD        STERNPLANE  STOP
STRPROP     SVDECOMP    TAG         TAIL        TAILEFF
TARGET      TEXT        THE         THRERRGAIN  THRUST
TIME        TIMEHISTRY  TIMESTEP    TOC         TOLERANCE
TORPEDO     TRATIO      TWIST       TWOPARAM    TYPEONE
TYPEZERO    U           UKT         UKTERRGAIN  UMS
UMSERRGAIN  UP          V           VEHICLE     VENT
VOLUME      VRML        W           WAGBINIT    WAGENINGB
WAKE        WEIGHTDCI   WHITE       WTC         YAW
ZERO
```

Annex C. Program Unit Lists

Build 061102 source code is organized in a series of Fortran files as follows. The program unit names are cited in error messages.

`Dssp21.for`; the highest level routines:

Unit	Function
`DSSP21`	main program
`GEO_L1`	process *Root*`.geo` level 1 input
`GEO_L2_AUTO`	process *Root*`.geo` autopilot level 2 input
`GEO_L2_HPA`	process *Root*`.geo` `HPAir` level 2 input
`GEO_L2_HUL`	process *Root*`.geo` `Hull` level 2 input
`GEO_L2_LFT`	process *Root*`.geo` `Lift` level 2 input
`GEO_L2_LMP`	process *Root*`.geo` `Lump` level 2 input (inactive)
`GEO_L2_MBT`	process *Root*`.geo` `MBT` level 2 input
`GEO_L2_OOP`	process *Root*`.geo` `OOPCalc` level 2 input
`GEO_L2_PLT`	process *Root*`.geo` `Plot` level 2 input
`GEO_L2_PRP`	process *Root*`.geo` `Propulsor` level 2 input
`GEO_L2_WIN`	process *Root*`.geo` `WagBInit` level 2 input
`GEO_L2_WTC`	process *Root*`.geo` `WTC` level 2 input
`INP_L2_REF`	process *Root*`.geo` and *Root*`.sim` `Reference` level 2 input
`SIM_L1`	process *Root*`.sim` level 1 input
`SIM_L2_CAP`	process *Root*`.sim` `Captive` level 2 input
`SIM_L2_COF`	process *Root*`.sim` `Coefficnt` level 2 input
`SIM_L2_FRE`	process *Root*`.sim` `Free` level 2 input
`SIM_L2_MLD`	process *Root*`.sim` `MLD` level 2 input (inactive)
`SIM_L2_STA`	process *Root*`.sim` `Static` level 2 input

`Sp21cint.for`; time simulation integration:

Unit	Function
`DDNORK`	order (3,4) Runge-Kutta integrator
`INTPC0`	fourth order interpolation, C^0 continuous
`INTPC1`	fourth order interpolation, C^1 continuous

`Sp21esam.for`; added mass estimation:

This file comprises the core subroutine `ESAMCR` and associated program units adapted, with only minor modifications, from the `ESAM` added mass program [6]. To avoid future

naming conflicts the program units are listed here: ESAMCR, AMMATX, AMOUTP, BCZB, CHGSGN, CHIR, COS2Z, INERCO, QKE, SWITCH, UEU, VFIELD, and ZEROIF. There should be no errors reported from within this set of routines.

Sp21prep.for; geometry and systems preprocessing:

Unit	Function
ALBON	hull axial potential flow using Albone's methods
ALBONC	hull axial potential flow for a hull with a closed tail
ALBONO	hull axial potential flow for a hull with an open tail
AXIBBL	hull axial flow thin boundary layer
HULLAX	check hull local axes specification
HULLGE	process hull geometry
HULLHY_HUL4	hull hydrodynamic calculations using the HulFor method
HULLH1_HUL4	hull linear damping from HulFor
HULLPT	save hull geometry definition for *Root*.gpl
LIFTDL	appendage orientation parameters
LIFTGE	process appendage geometry
LIFTHY	appendage hydrodynamic calculations
LIFTLO	appendage local axis calculations
LIFTPT	save appendage geometry definition for *Root*.gpl
PLOTGE	manipulate geometry for plotting
PLOTXA	write *Root*.gpl in AcroSpin format
PLOTXM	write *Root*.gpl in MatLab format
PLOTXP	write *Root*.gpl in Maple format
PLOTXV	write *Root*.gpl in VRML 1 format
PLOTXW	write *Root*.gpl in VRML 2 format (inactive)
RUDCOF	Bohlmann-Spreiter method for tailplane efficiency
SPREIT_WB	Spreiter's loading function for K_{WB}
SPREIT_W_B	Spreiter's loading function for k_{WB}
THWAIT	hull axial laminar boundary layer — Thwaite's method
TRUCK	hull axial turbulent boundary layer — Truckenbrodt's method
WAGBINIT	initialize Wageningen B propellers

Sp21sims.for; simulation run calculations:

Unit	Function
AUTOCOM	set up parameters for autopilot commands
AUTODOIT	update autopilot error and control signals
AUTOFIND	process autopilot command records
BLOCOM	interpret Blow commands

BLOWVENT	update BG and MBT data during a ballast blow or vent
COEF_D	hydrodynamic coefficient calculation with the DERIVS method
COMAND	process free simulation command records
D_ABCD	three/four point coefficient evaluation for COEF_D
DYDT	define the extended equations of motion, $\dot{\mathbf{u}} = F(\mathbf{u}, t)$
FLOOD	update WTC data during a flood
FLUDCOM	interpret Flood commands
HEADFIND	process heading argument strings
SPCOUT	output free simulation to sim*nnn*.txt and aux*nnn*.txt
SPRINT	execute a free simulation sprint
STABILITY	small perturbation stability analysis
TIMHXN	estimate loads for a captive time history
VENTCOM	interpret Vent commands

Sp21subs.for; general subroutines:

Unit	Function
ADDMGEN	generate component added masses
ADDMMAT	setup added mass matrix in reference axes
CANINC	calculate floodwater volume and centroid
CANMOX	function for floodwater longitudinal moment
CANMOZ	function for floodwater vertical moment
CANVOL	function for floodwater volume
CFTURB	turbulent flow friction drag coefficient
CFWET	estimate hull friction drag
DELASSOC	set up control appendage Delta command associations
DELCOM	process Delta commands
DELFIND	interpret Delta command strings
DELLIM	reset control surface deflection limits
DYNAMV	estimate dynamic loads for the complete vehicle
FLUDCG	estimate floodwater centroid as a function of volume and pitch
HULL_LOOK	in-plane hull load lookup tables
HYDRST	rationalize **BG**, **CB**, and **CG**
INRTMAT	setup inertial matrix in reference axes
IWEIGHTS	process WeightDCI records
KTKQO2	DSSPtwo propulsor model
KTKQST	STRprop propulsor model
KTKQWB	WageningB propulsor model
NLOCGO	save local axis systems in vehicle axes
NREFGO	adjust local axis systems to reference axes
OOPGRADE	update out-of-plane time histories

OOPSET	static out-of-plane calculation
OOPSTEP	dynamic out-of-plane calculation
PRPCOM	process Propulsor command records
PRPFRC	calculate propulsor loads
PRPINT	initialize propulsion condition
PRPINX	estimate initial propulsor loads
RATCOM	rationalize simulation commands and conflicts
REVPPT	process CB, CG, or Reference redefinition
STATHL	Hull static forces and moments
STATHL_HUL4	HulFor static load calculation
STATLM	Lump static forces and moments (inactive)
STATLT	Lift static forces and moments

Sp21util.for; low level utilities:

Unit	Function
ADDRNG	add vector to a double precision ring array
CHCOMP	word string comparison
CIRCIH	returns angle in range $-\pi$ to π
CIRCIT	returns angle in range 0 to 2π
COMP3F	compare number triplets
CROOTS	obtain roots of cubic equation
DECOMP	matrix decomposition by Gaussian elimination and partial pivoting
DEFLBL	initialize the component label array
GETCHR	get the next word in the input stream
GETDBL	convert the next word in the input stream to double precision
GETINT	convert the next word in the input stream to integer
GTDRNG	get vector from a double precision ring array
HELP	write program information to *Root*.log
HELP_INP	write *Root*.geo and *Root*.sim input summaries to *Root*.log
INDICT	find dictionary keyword matching a character string
INDRNG	insert vector into a double precision ring array
INLABL	find component label matching a character string
INTRPL	interpolation of a single valued tabulated function
LOOKUP2	lookup table bilinear interpolation
MESSAG	error message manager
NEARNESS	distance from a point to a line
OPENIT	open a file with status checks
PCQUAD	quadrature using piecewise cubics

PYTHAG	robust solution of $\sqrt{a^2 + b^2}$
QSFM	fourth-order quadrature, for evenly-spaced abscissae
SOLVE	solve linear system of equations decomposed by DECOMP
SVBKSB	solve linear system of equations decomposed by SVDCMP
SVDCMP	matrix singular value decomposition (SVD method)
TIQUAD	return coefficients for a three-point quadratic fit
TOLABL	add an entry to the component label list
UCCHAR	character case conversion and filtering
WORDIN	find a word within a character string

In addition, the following 'Include' files are required to define common blocks and shared parameters:

```
Sp21aaaa.cmn
Sp21adel.cmn
Sp21aioa.cmn
Sp21aplt.cmn
Sp21auto.cmn
Sp21ggeo.cmn
Sp21labl.cmn
Sp21stat.cmn
Sp21stuf.cmn
```

Annex D. Example Root.geo Input File

This is a fairly typical input file for a full scale submarine. However, for purposes of illustration more of the optional inputs have been used than are usually necessary.

```
Text  Demo
Text  68 m generic submarine with a deck and sailplanes


Reference
 -30.2493 0.  0.   !  Reference at the (precalculated) hull
   0.     0.  0.   !           CB, 0.444842*ell aft of nose


BG 0.38 ! Form CB used for vehicle CB, with CG displaced from it by BG


Hull               ! Default type and number of stations
   Label #Hull
   Station                                ! Regular, 21 stations
     0.0000  0.0000   0.0000  0.0000     ! Camber is cosmetic
     6.2499  6.2166  30.5152   .0170
     7.3481  7.5364  43.4940  -.0945
     7.7030  8.1695  49.4248  -.2332
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.7717  8.3545  50.9949  -.2917
     7.6500  8.1892  49.1731  -.2700
     7.2012  7.5779  42.8592  -.1884
     6.4219  6.5443  33.0078  -.0612
     5.3122  5.3122  22.1635  0.0000
     3.8719  3.8719  11.7744  0.0000
     2.1012  2.1012   3.4676  0.0000
     0.0000  0.0000   0.0000  0.0000
   Nose   0.0  0.0  0.0                ! Vehicle origin at nose
   Tail -68.0  0.0  0.0
   LoD    8.75
   Source NeuralNet                   ! Most hull parameters take
   Deck   1.2  1.05                   !        the default values
```

```
Lift  Sail
   Label #Sail
   Sail  4.1772
   Planform
   -20.4000 0.  -4.4690
   -20.4000 0. -10.5468
   -32.0552 0. -10.5468
   -32.0552 0.  -4.4690
   ToC    0.2
   CXJ   -0.12                  ! Sail junction drag estimate


Lift
Label #StarboardSternplane
   Tail  2.1733  3.8855
   Flapped  0.0  1.0  0.5660
   Planform
     -58.8438  2.4528  0.0
     -60.2290  4.8606  0.0
     -64.1145  4.8606  0.0
     -64.1145  1.1873  0.0
!!   TailEff    dempsey          ! Commented out; will use default
   FlaProDrag  0                 ! Ignore flap profile drag
   ToC  0.15
   DelRateLim   5.0
   Save

Lift
   Label #PortSternplane
   Rotate 180                   ! Stbd plane rotated 180 deg
Lift
   Label #UpperRudder           ! Rotated sternplane - all tail appendages
   Rotate 270                   !                            are identical
Lift
   Label #LowerRudder
   Rotate 90                    ! As above

Lift
   Label #StarboardSailplane
   Sailplane 1.1655 .4125
   AllMoving 1.0
   Planform
     -22.4400 1.0819  -7.2163
     -23.4600 3.8855  -7.2163
     -24.1400 3.8855  -7.2163
     -24.1400 1.1642  -7.2163
   ToC 0.15
   DelRateLim   5.0
   WeightDCI    0.  0.  0.      ! Decoupled
   Save
```

```
Lift
   Label #PortSailplane
   Reflected                     ! Stbd plane reflected

Plot                             ! Use default - VRML

OOPcalc
   Lift #Sail
   KParam -1                     ! Force k = -1

Prop
   Location    -65.9600          ! Propulsor hub (on hull axis)
   Right Handed                  ! Redundant; 'Handed' is ignored
   Diameter      3.9090
   RevLimit    220.0
   RevRateLim   11.0

WagBInit
   Diameter   7.6                ! Hull diameter for prop initialization


AutoDepth   ! Autopilot parameters have not been optimized for this case
   LookAhead    68.0             ! Lz = L

AutoHead
   HedErrGain    1.0             ! (1/deg)

AutoProp
   UmsErrGain  5.0               ! (1/(m/s))


Text Ballast system organized like figs 8 and 15 in the user guide

HPair
   Label #BGOne
   Fwd
   Pressure    260.
   Volume        2.0

HPair
   Label #BGTwo
   Main
   Pressure    200.
   Volume        3.4

HPair
   Label #BGThree
   Aft
   Pressure    300.
   Volume        1.1
```

```
MBT
   Label #TankOne
   Fwd
   Location     -4.0   0.  1.2
   Volume       70.0
   MassFlow     14.0
   Vent         10.0
MBT
   Label #TankTwo
   Fwd
   Location     -8.0   0.  1.0
   Volume       50.0
   MassFlow     12.0
   Vent          6.0

MBT
   Label #TankThree
   Aft
   Location    -58.5   0.   0.8
   Volume       48.
   MassFlow     11.2
   Vent          6.0

WTC                             ! Only one input
   Label #MidCompartment
   Diameter    7.72
   Length     23.8
   Location  -32.3 0 0
   Occupancy   0.22
```

Annex E. Example Root.sim Input File

This is not a typical input file; it is somewhat more elaborate than usual in order to illustrate various input options. For housekeeping purposes it is preferable not to mix so many run types on a single input file.

```
Free            ! Sim 1

   Text 15/15 zigzag with manual depth control using sailplanes:
   Text target is +/- 1 m over the duration of the sim.  Both
   Text depth control and rudder timing found by trial and error

   Rho  1025.2
   Depth 100.
   Ukt    10.

     0.    Del port   15.
    16.7   Del port  -15  ! reverse rudder at 15 deg port heading...
    50     del 6 -8
    50     del 7  8
    55.7   Del port   15. ! ... and so on.
    80     del 6 -4
    80     del 7  4
    96.3   Del port  -15
   105     del 6 -7
   105     del 7  7
   137.3   Stop


Static          ! Sim 2

   Text Standard pitch sweep as in a model test
   Loop Alfa -30 30 2          ! Note Rho is still 1025.2


Coef            ! Sim 3

   Delta
     Starboard           ! Use associated types with current weights
     Down                ! - as above
     WeightDCI All Zero  ! Zero all current weights
     WeightDCI 6  1 0 0  ! Turn on stbd sailplane,
     WeightDCI 7  1 0 0  !   turn on port sailplane,
     Down                !   and do for them alone
     WeightDCI All Reset ! Back to baseline weights
   Output coef
   PReverse    -62.7968  ! Sternplane root quarter-chord
```

```
Captive         ! Sim 4

   Rho       1022.               ! Redefine Rho
   DEL #PortSternplane      -5   ! Initialize sternplanes
   DEL #StarboardSternplane  5   !     "          "
   TimeHistry  tinyhist.txt      ! Example in section 21.2


Free            ! Sim 5

   Text A flooding scenario; however, this is far from a standard
   Text recovery procedure, but is used to illustrate various
   Text options in the program.  Note that vertical control
   Text authority is initially poor -- see sternplane reversal
   Text speed estimate in sim 3.

!(Several features of this -- including use of autopilots and
! blowing and flooding -- make this run slower than sim 1 by a
! factor of about 20.  Reducing the default integration tolerance
! by an order of magnitude:
   Tolerance   0.01
! mitigates the problem to a factor of about only 1/3 with no
! significant impact on key run time histories such as depth
! and pitch.  However, such a change should always be tested on
! representative runs in a series before applying it widely.)

   Rho  1025.2                   ! Redefine Rho
   Depth  65.
   Heading East
   Ukt     4.
   Interp  1.                    ! Interpolate with a 1 sec interval

     0.  Start                   ! Unnecessary -- the next will start it
     0.  Autodepth 110.          ! Depth change to 110 m,
     0.  Prop 78.                ! Increase speed for control authority
   158.  Flood #mid 0.08         ! Flood starts as depth passes 90 m.
   195.  Autodepth Stop          ! Start recovery
   197.  del up 25               ! Planes manually to rise
   209.  del up  5               ! Ease off planes as pitch passes zero
   209.  Blow 2 1                ! Get some buoyancy
   219.  Blow stop               ! Secure blow after 10 s
   240.  Flood Stop              ! Secure flood
   250.  Vent 2                  ! Dump additional buoyancy
   250.  Prop 104                ! Speed to approx 8 kt
   250   Del Up 15               ! Rise
   270   Autodepth  20           ! Hold depth at 20 m
   480.    Stop
```

Nomenclature

Symbols

$\mathbf{A}$	added mass matrix
A_B	hull base area
A_i	hull station area
b	semispan of isolated appendage, or measured from hull centerline
b_{ep}	spanwise location of endplate
b_{exp}	exposed appendage span
$b_{sp(exp)}$	spanwise location of sailplanes on exposed sail
b_1, b_2	spanwise location of inboard and outboard flap extent
B	buoyancy force
B_i	hull station breadth
$\overline{BG}$	metacentric height
$\mathbf{BG}$	metacentric distance vector
$c, \overline{c}$	appendage chord, mean chord
$\overline{c_f}$	mean flap chord
ci, $i = 1, 4$	appendage planform corner points
$\mathbf{C}_i$	loads on vehicle component i
$\mathbf{C}_{i,j}$	interaction loads for vehicle components i and j
$\mathbf{CB}$	center of buoyancy vector
C_C	autopilot control gain
C_E	autopilot error gain
C_F	flooding coefficient of discharge
$\mathbf{CG}$	center of gravity vector
C_X, C_Z	axial and normal force coefficients
C_{Xj}	junction axial force coefficient
C_{X0}	appendage zero lift axial force coefficient
$C_{X\alpha\alpha}$	appendage induced drag axial force coefficient
$C_{Z\alpha}$	appendage linear normal force coefficient
$C_{Z\alpha\alpha\alpha}$	appendage nonlinear (crossflow drag) normal force coefficient

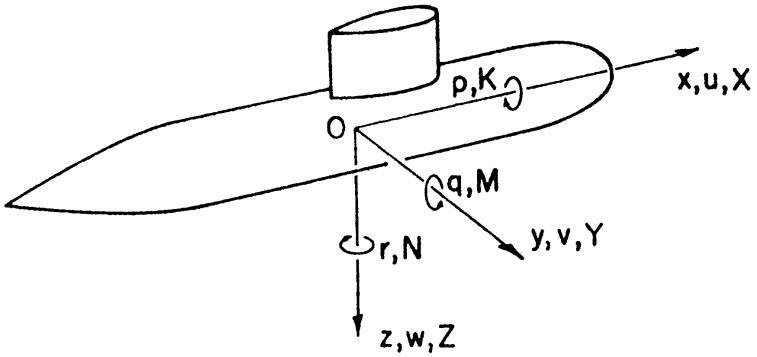
$C_{\Delta f}$	hull residual drag coefficient
D_C	diameter of WTC
D_F	flooding breach diameter
D_M	maximum hull diameter
D_P	propeller diameter
f	appendage camber
$\mathbf{F}$	hydrodynamic load vector
F_i	hull station camber
h	total endplate height, or integration timestep
h'	endplate height from attachment
h_{min}, h_{max}	lower and upper limits of timestep h
$\mathbf{I}$	inertial matrix
$I_x, \ldots, I_{xy}, \ldots$	moments and cross products of inertia
J_C	autopilot control signal
J_E	autopilot error signal
J_S	advance coefficient based on vehicle speed
k	out-of-plane load distribution parameter
k_1, k_2	PID autopilot error damping equation coefficients
k_r	hull roughness allowance
k_{WB}	tailplane efficiency for a control deflection
k_δ	appendage flap profile drag
k_ϕ	appendage anhedral factor
K, M, N	roll, pitch, and yaw moment
K_P, K_I, K_D	PID autopilot coefficients
K_T	propulsor thrust coefficient
K_Q	propulsor torque coefficient
K_{WB}	tailplane efficiency for incidence
ℓ	reference length for nondimensionalizing, typically LBP
l_z	depth autopilot lookahead distance
L	hull length
L_C	length of WTC
m	mass
$\dot{m}_{BT\bullet}$	initial mass flow rate of air blowing into MBT
$\dot{m}_{BV}$	mass flow rate of air venting from MBT

n	propulsor RPM
n_M	maximum RPM
$\dot{n}_M$	RPM rate limit
n_{segH}	number of hull cross section segments for plotting
n_{segL}	number of appendage cross section segments for plotting
$n_{version}$	VRML format version number
n_0	self-propulsion RPM
$n_\bullet$	initial RPM
N_B	MBT index
N_C	WTC index
N_G	HP air BG index
N_H	hull index
N_{iter}	iteration limit for estimating `Captive` Euler angle time histories
N_L	lifting appendage index
N_P	propulsor index
N_S	number of hull stations for load calculation
N_{sep}	station for hull separation
$N_{S_{in}}$	number of hull stations input
O_C	WTC occupancy
p, q, r	roll, pitch, and yaw rate
$P_{HP_\bullet}$	initial HP air BG pressure
P_P	propeller pitch
Q	arbitrary quantity or parameter
r	local hull radius
r_M	maximum hull radius
r_{view}	viewing distance for VRML plots
R_c	chord Reynolds number
S_{exp}	exposed appendage area
S_m	area of moving part of all-moving appendage
t	time, or appendage thickness
Δt	time step
T_i	hull station height
$\widetilde{T}_i$	thrust ratio for i^{th} propulsor
u, v, w	axial, lateral, and normal velocity

$\mathbf{u}$	state vector
U	total velocity
$U_\bullet$	initial forward speed
V_{BT}	MBT volume
V_{HP}	HP air BG volume
W	weight
W_{ID}, W_{IH}, W_{IR}	weights for indirect control of depth, heading, and roll
x, y, z	vehicle axis coordinates, local axes
x'	distance downstream from origin of hull boundary layer
x_0, y_0, z_0	coordinates in earth-fixed axes
x_B, y_B, z_B	center of buoyancy coordinates
x_{BG}, y_{BG}, z_{BG}	components of **BG** vector
x_{BT}, y_{BT}, z_{BT}	coordinates of MBT location
x_C, y_C, z_C	coordinates of WTC location
x_{ci}, y_{ci}, z_{ci}, $i = 1, 4$	appendage corner point coordinates
x_e, y_e, z_e	chart axes
$x_{e\bullet}, y_{e\bullet}$	initial chart position
x_G, y_G, z_G	center of gravity coordinates
x_{in_i}, $i = 1, N_{S_{in}}$	hull station axial coordinate
x_m, y_m, z_m	hull midpoint coordinates
x_n, y_n, z_n	hull nose coordinates
x_o, y_o, z_o	vehicle reference origin
x_P, y_P, z_P	propulsor location coordinates
x_t, y_t, z_t	hull tail coordinates
x_T, y_T, z_T	target point coordinates
$x_\delta, y_\delta, z_\delta$	control surface location
X, Y, Z	axial, lateral, and normal forces
$z_{0\bullet}$	initial depth
α	angle of attack, or phase lead autopilot amplitude gain
α_b	appendage geometric twist
$\alpha_{(1)}, \alpha_{(2)}, \alpha_{(3)}$	equivalent angles of attack for local appendage forces
β	angle of drift
δ	control surface deflection
δ_e	effective flap angle of deflection

$\delta_{Max}, \delta_{Min}$	maximum and minimum control deflection angles
$\dot{\delta}_M$	control deflection rate limit
$\delta_\bullet$	initial control deflection
ϵ	time integration tolerance
ϵ_{iter}	tolerance for estimating `Captive` Euler angle time histories
ϵ_{SVD}	tolerance for SVD matrix inversion
ϵ_ω	tolerance for `Captive` Euler angle consistency
ζ_n	RPM rate damping
ζ_δ	control deflection damping
η	appendage flap gap correction
η_{Yd}, η_{Nd}	deck sideforce and yawing moment corrections
η_{Yk}, η_{Nk}	keel sideforce and yawing moment corrections
θ	pitch angle
θ_L	depth autopilot pitch limit
θ_P	propulsor local axis pitch
Θ	axis rotation in pitch
λ_1, λ_2	hull-bound vortex distribution parameters
ρ	water density
τ	phase lead autopilot anticipation time constant
ϕ	roll angle, or appendage anhedral or dihedral angle
ϕ_o, θ_o, ψ_o	vehicle reference axis directions
ϕ_R	angle of rotation for a copied appendage
Φ	axis rotation in roll
ψ	yaw angle, heading
$\psi_\bullet$	initial heading
ψ_c	commanded heading
ψ_P	propulsor local axis yaw
Ψ	axis rotation in yaw
ω_n	RPM rate frequency
ω_δ	control response frequency
Ω	vector angular velocity

Reference Axes and Loads



Acronyms and Abbreviations

AIP	Air Independent Power
BG	(HP air) Bottle Group
CB	(vehicle) Center of Buoyancy
CF	Canadian Forces
CFD	Computational Fluid Dynamics
CG	(vehicle) Center of Gravity
EOR	End Of Record
GUI	Graphical User Interface
HP	High Pressure
LBP	Length Between Perpendiculars
MBT	Main Ballast Tank
MLD	Manoeuvering Limitation Diagram
ODE	Ordinary Differential Equation
PID	Proportional-Integral-Derivative (control)
ROV	Remotely Operated Vehicle
SVD	Singular Value Decomposition
UUV	Unmanned Underwater Vehicle
VRML	Virtual Reality Modeling Language
WTC	WaterTight Compartment

DOCUMENT CONTROL DATA

1. ORIGINATOR	2. SECURITY CLASSIFICATION	
Defence R&D Canada – Atlantic	Unclassified	
3. TITLE		
DSSP21 Build 061102 User Guide		
4. AUTHORS		
M. Mackay		
5. DATE OF PUBLICATION	**6a. NO. OF PAGES**	**6b. NO. OF REFS**
December 2006	117	26
7. DESCRIPTIVE NOTES		
DRDC Atlantic Technical Memorandum		
8. SPONSORING ACTIVITY		
9a. PROJECT OR GRANT NO.	**9b. CONTRACT NO.**	
11GP02		
10a. ORIGINATOR'S DOCUMENT NUMBER	**10b. OTHER DOCUMENT NOS.**	
DRDC Atlantic TM 2006–252		
11. DOCUMENT AVAILABILITY		
Unlimited		
12. DOCUMENT ANNOUNCEMENT (if different from 11)		

13. ABSTRACT

DSSP21 is a semiempirically based computer code for evaluating the dynamics and manoeuverability of streamlined underwater vehicles including submarines. The purpose of this user guide is to facilitate preparation of the input files that are required for a definition of the vehicle and its systems, and for controlling the calculations and manoeuvering simulations to be done. It contains detailed descriptions of input syntax and options, together with brief discussions of the output produced and some background information on the code structure and algorithms.

14. KEYWORDS

Underwater Vehicles
Submarines
Manoeuvering
Simulation

This page intentionally left blank.

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca